Document Image Analysis and Text Recognition on Khmer Historical Manuscripts

A thesis submitted in partial fulfillment of the requirements for the degree of Docteur en sciences de l'ingénieur et technologie, Université catholique de Louvain

January 2020

DONA VALY

Thesis committee:

Prof. Michel Verleysen (UCLouvain), supervisor
Prof. Jean-Pierre Raskin (UCLouvain), chairperson
Prof. John Lee (UCLouvain), secretary
Prof. Bernard Gosselin (UMons, Belgium)
Dr. Sophea Chhun (ITC, Cambodia)

List of Figures

List of Tables

Abstract

Palm leaves have been used as one of the major sources of writing and painting in many Southeast Asian countries. In Cambodia nowadays, palm leaf documents called “Sleuk Rith” in Khmer are still around attributable to their cultural value as well as the precious contents written on them. However, as a consequence of deterioration from natural aging and damage caused by various natural factors, palm leaf manuscripts are facing destruction and are in need for preservation. Many programs and projects are underway to recover and preserve palm leaf documents not only in their physical form but also in digital imaging through scanning and photography. The centralization of the digitized images allows easy access for the public. Nonetheless, searching and filtering the content of those documents using particular keywords are still unmanageable. An automatic recognition system therefore needs to be developed.

This dissertation takes part in exploring document image analysis (DIA) researches which put Khmer palm leaf manuscripts into the spotlight. We aim to bring added values by designing tools to analyze, index, and access quickly and efficiently to the text content of palm leaf documents. In order to achieve this objective, different DIA tasks are studied, and novel approaches to solve such tasks are proposed. First, a new corpus of digitized Khmer palm leaf manuscripts has been collected. From this corpus, the first Khmer palm leaf manuscripts dataset called “SleukRith Set” consisting of different types of annotated data has been constructed. Experimental evaluations and comparisons of approaches on various DIA tasks such as binarization, text line segmentation, and isolated character recognition have been conducted on Khmer palm leaf manuscript datasets in addition to datasets of palm leaf manuscripts from Indonesia. Moreover, we propose an efficient line segmentation scheme for grayscale images of Khmer ancient documents which is able to adapt to the curvature of the actual text lines and to produce separating seams using a path finding technique. We also introduce a novel concept of utilizing the annotated information of glyph components in the word image to build a glyph-class map

followed by a complete text recognition scheme using encoder-decoder mechanism. A new annotated data called “sub-syllable” which can be used as an efficient data augmentation technique for the text recognition task has been added to SleukRith set.

Acknowledgements

This dissertation would not have been possible without the help of many people who in one way or another contributed and extended their precious assistance to me during my PhD.

First and foremost, I would like to express my sincere gratitude to my advisor Prof. Michel Verleysen for his continuous support, patience, motivation, and enthusiasm. I am grateful for his immense knowledge in the field of machine learning, for his confidence in my potential, as well as for the perfect balance of guidance and freedom that he gave me to pursue my goals. I could not have imagined having a better advisor and mentor.

My appreciation to the head of my research project Dr. Sophea Chhun for her leadership and her help in dealing with challenges and in managing research activities throughout the whole project period. I would like to thank also Prof. Bernard Gosselin, member of my thesis committee, for his encouragement and insightful comments during my thesis confirmation.

Many thanks to Dr. Made Windu Antara Kesiman, Prof. Jean-Christophe Burie, and the team from Universitas Padjadjaran for their collaboration and for providing sample images of Balinese and Sundanese palm leaf manuscripts which are very valuable for the experimental studies of my research.

Last but not least, very special thanks go to my family and friends especially my mother, Phally, and my brother, Dyda for their constant love and support that they have never failed to offer.

This research study is supported by the ARES-CCD (program AI 2014-2019) under the funding of Belgian university cooperation.

Table of Contents

1. Introduction

1.1. Motivation

Dating back to centuries ago, in many countries in South Asia and Southeast Asia, palm leaves have been used as writing materials. Initially, knowledge was passed down orally, but after the diffusion of Indian scripts throughout most of Asia, people eventually began to write it down in dried and smoked treated palm leaves. These palm leaves have been used to record narratives from actual events or mythical phenomenon influenced by cultural and religious activities during those periods. The content of these documents are mostly handwritten texts, but some of them also contain drawings in black and white or even in color. With the spreading of Indian culture to Southeast Asian countries such as Thailand, Indonesia, and Cambodia, these nations became home to large collections of documents written on palm leaves.

In Cambodia, palm leaf documents are still seen in Buddhist establishments, mostly Buddhist temples called “pagodas”, and are considered to be holy and sacred. They are being used habitually and traditionally by monks as reading scriptures. Besides their cultural merit, these ancient manuscripts store valuable content including old religious sayings, crucial historical events, as well as certain scientific findings that are still very useful for researchers in those fields of study.

Nowadays, some programs to collect, preserve, and digitize palm leaf documents are under way, but only few have been completed especially in Indonesia and Cambodia. The main goal of these efforts is to preserve cultural heritage embedded with these ancient documents before they face destruction due to their fragility. From previous preservation campaigns, the manuscripts are scanned empirically, sometimes with poor resolution and inadequate light by different stakeholders in diverse institutions (public or private). Moreover, the storage of the digital files is not always safe, lacking backup and mirror storage.

Beside the objective of preserving physical conditions of the manuscripts i.e. preventing damage from the injuries of time (dirt,

moisture, insects) and accidents (fire, floods), these programs aim also at facilitating the access to the content of the documents for researchers. Currently, most of the digitized images are in reality not accessible to the public. Remarkable efforts in providing open access to the data along with a precise bibliometric indexing have been made in Cambodia, for instance the collaborative work with Ecole Française d'Extrême-Orient (EFEO) which has made available a database[1] of thousands of collections of palm leaf manuscripts digitized from microfilms. This excellent example of open-handed data is unfortunately still rare. Nevertheless, to date, only bibliometric indexing can be offered beside the image files: identification of the manuscript origin, title, date, topic, etc., but no indexing of the content is available. Therefore, in-text search by keywords is still impossible. As far as we know some work has been done to enhance the quality of images of palm leaf manuscripts but no research to analyze and to index automatically the content of these ancient documents has been conducted. People, such as historians or philologist wanting to study these ancient documents have to read them one by one to find the needed information, and they have to go, most of the time, physically to the place where the documents are stored. To enable this in-text search capability as well as applications related to text processing on palm leaf manuscripts, a text recognition system is therefore needed.

1.2. Objectives

This research work aims at bringing added value to digitized palm leaf manuscripts by developing tools to analyze, index, and access quickly and efficiently to the content of ancient manuscripts. A document image analysis (DIA) system is to be developed. The development of the system is divided into several step-by-step image processing tasks starting from the construction of digital corpus and ground truth data. Next, the pre-processing phase includes analyzing document layouts and segmenting document images into smaller entities such as text lines and individual words or syllables. Finally, the segmented elements will

[1] http://khmermanuscripts.efeo.fr

be recognized and transformed into output texts to complete the text recognition scheme.

Under the scope of this research work, experimental study is performed on palm leaf manuscripts from Cambodia and Indonesia; however, documents written in Khmer script are the main focus. With respect to this type of documents, many technical challenges will be assessed. A standardized methodology for safe and efficient manuscript digitization will be adopted in order to preserve the physical integrity of the manuscripts and to ensure a necessary and proper quality to analyze the document. Secondary data such as old photographs of palm leaves manuscripts which can be found in libraries and various institutions will also be taken into account. The DIA system will therefore be able to deal with these palm leaf documents under a variety of qualities. The palm leaves are sometimes in poor physical conditions, and although they are digitized properly, scratches and artifacts might still occur in the images. The developed DIA system should be able to eliminate most of those noises and to extract only the relevant information. Other challenges including unique characteristics of these palm leaf documents such as their format and layout as well as the complexity of the script written on them (Chapter 2.3) should also be considered.

After the completion of the DIA system, an interactive search engine is expected to be developed. The idea is to propose a tool allowing to search all the instances of a specific word inside the corpus. Thanks to this search engine, rather than spending extensive amount of time searching for some specific content, researchers (philologists and historians for example) will be able to access quickly to the relevant information.

1.3. Contributions

The main contributions of this thesis are listed as the following (the presented work has been published in large parts in various international conference papers and journals which are also referenced here).

1) **Ground truth creation:** A collection of digitized Khmer leaf manuscripts has been collected to create a corpus. A tool has been designed and developed to specifically build the ground truth consisting of different types of annotated data suitable for experimental studies on various DIA tasks. The first Khmer palm leaf manuscripts dataset called "SleukRith Set" has been constructed and made available to public [1].
2) **Benchmarking of DIA methods on palm leaf manuscript datasets:** Experimental evaluations and comparisons of approaches on various DIA tasks such as binarization, text line segmentation, and isolated character recognition have been conducted on Khmer palm leaf manuscript dataset in addition to datasets of palm leaf manuscripts from Indonesia [2, 3].
3) **Line segmentation:** A novel text line segmentation approach on binary images has been proposed [4]. The approach starts by determining the number of lines and setting up text line mid points' initial positions using a modified piece-wise projection profile technique. The competitive learning algorithm is applied afterwards to adaptively move those mid points according to the geometrical information of connected components in the document page to form lines. Borders between text lines are defined so that they can be used to separate touching components that spread over multiple lines. The proposed method is robust in handling documents with skewed, fluctuated, or discontinued text lines. An extended binary-free version of the approach has also been proposed i.e. the approach works directly on grayscale or color input images [5].
4) **Character and text recognition:** trial experimentations on isolated character recognition as well as text recognition on word image patches have been conducted using different types of neural network architectures such as Convolutional Neural Networks, Long Short-Term Memory Recurrent Neural Networks, and a combination of both [6]. A novel concept utilizing the annotated information of glyph components in the word image to build a glyph class map is introduced followed by a complete text recognition scheme using encoder-decoder

mechanism [7]. A new annotated data called “sub-syllable” has been added to SleukRith Set along with a new optimized text recognition system adapted on this data. This new data can be used as an efficient data augmentation technique for the text recognition task.

Not included in this manuscript are related collaborative works of organizing competitions on DIA tasks for Southeast Asian palm leaf manuscripts for a wider research community in the 15th and 16th International Conference on Frontiers in Handwriting Recognition (ICFHR 2016 and 2018) [8, 9]

1.4. Outline

The remainder of this dissertation is organized as follows. Chapter 2 presents a detailed description of Khmer palm leaf manuscripts. This chapter includes a brief introduction to the history of palm leaf manuscripts in Southeast Asia followed by the process of transforming raw palm leaves into writing medium as well as a presentation about Khmer language, its script, and its writing style. DIA challenges from the conditions and characteristics of Khmer palm leaf documents and from the complexity of Khmer script are also discussed. Next, Chapter 3 focuses on literature review which examines state-of-the-art approaches on various DIA tasks such as binarization, segmentation, and text recognition system which is subdivided into the traditional handcrafted feature extraction approaches and the deep learning methods. The collection of digital corpus of Khmer palm leaf manuscripts, the construction of ground truth tool and data, and the introduction to Sleuk Rith Set, the first Khmer palm leaf manuscript dataset, are presented in Chapter 4. After that, Chapter 5 first presents a benchmarking and comparison study of binarization approaches from the literature. This chapter also introduces the proposed binary-free line segmentation method as well as experimental evaluation with some base-line approaches. In Chapter 6, we cover the initial experimentations on using different neural network architectures on solving individual character recognition as well as the task of localizing glyphs in short image patches. After the feasibility study of using deep

learning approaches for Khmer text recognition problem, in Chapter 7, we present an overview of the proposed text recognition scheme. Finally, conclusions are drawn, and future research directions are discussed in Chapter 8.

2. Khmer Palm Leaf Manuscripts

In this chapter, details on Khmer palm leaf manuscripts are discussed. A general description, including a brief history and the production process of palm leaf documents, is presented. Khmer language whose script is handwritten on the manuscripts is also introduced. At the end of the chapter, we look at DIA challenges from two aspects of historical Khmer palm leaf documents: (1) the conditions as well as certain characteristics of such documents and (2) the complexity of Khmer script.

2.1. General Description of Palm Leaf Manuscripts

The concept of writing has been revolutionized throughout many civilizations. The main purpose of writing is to be used as an alternative form of communication and also to record important information. Before the invention of paper, the geographical condition of each nation was the main influence on the choice of natural materials to be used as a medium for writing.

2.1.1. A Brief History

Palm leaf manuscripts are one of the oldest mediums and also one of the major sources of writing and painting in Southeast Asian countries including Nepal, Sri Lanka, Burma, Thailand, Indonesia, and Cambodia [10]. Although the practice of palm leaf writing existed since the ancient times, it is still unclear about its precise origin. According to [11], it is difficult to say exactly when palm leaves began to be used for writing; however, palm leaves were definitely in use much earlier than the 10th century since it is mentioned as a writing material in several literary works and its visual representation can be seen in several sculptures and monuments. Until today, the composition and method of writing of palm leaf manuscripts have remained unchanged. Even though not as high in volume, people still prepare and use palm leaf manuscripts the way our ancestors used centuries ago.

2.1.2. From Palm Leaves to Writing Materials

Due to the natural size of the leaves, palm leaf manuscripts are always found in linear horizontal format. Each palm leaf page is long and

rectangular in shape whose dimension varies from $15cm$-$60cm$ in width and from between $3cm$-$12cm$ in height (Figure 2.1). Their dimension normally depends on the available size of the leaves. To make them suitable for scribing, before writing the palm leaves need to be processed and prepared.

2.1.2.1. Processing of Palm Leaves

Palm leaves are first selected from the tree and cut into required size. Only half open young shoots of palm leaves are suitable for making manuscripts. To avoid being too dried up and becoming brittle, the cut palm leaves are placed in an organized manner to dry (Figure 2.2[2]). They need to be taken very good care of, or they will be damaged, torn, and saturated. The drying process can be done at early dawn or during the night to take advantage of the dew which can make the leaves soft. After the stems are extracted from the dried leaves, they need to be wiped to remove dirt. When they are all clean, we arrange and tie them by putting them in a splint in order to be pressed tightly together. This process takes up to two or three months so that the leaves can be straightened properly. As a final stage of the preparation process, the earlier pressed leaves are smoked and roasted to remove remaining humidity. The leaves are finally ready for writing.

Figure 2.1: A collection of Khmer palm leaf manuscripts

[2] From the documentary "Project Three 2016: Sleuk Rith, My Life" produced by Ream Chamroeun, Leng Sreynich, Seng Vannak, and Ngy Sovan Ratany

Figure 2.2: Palm leaves are being placed to dry

2.1.2.2. Writing on Processed Palm Leaves

Writing on palm leaves is a skilled activity which requires patience, practice, and training. Before writing, the readily prepared leaves are marked using thin threads to create lines. Normally five rows of lines sit on both the back and the front of the leaf. Incision with a sharp metal stylus is the most common method of writing on palm leaves. The text to be written is therefore carved on the leaf one letter at a time. After the incision, the letters may not be visible to read. A mixture of dark burned wick from oil lamp and wood oil, often called "Mrenh Plerng", is used to rub on the leaf so that the engraved letters become more noticeable and easier to read. The excess mixture is then wiped off with a cloth (Figure 2.3). Since correction or overwriting is difficult, great attention is required to make each leaf error free.

After making sure that all texts are correctly inscribed, the pages are then secured between two wooden panels that are slightly larger in size than the leaves. To keep the leaves together, holes are punched in the them. One hole is normally created in the center if the leaf is small otherwise two holes are punched at either end of it. A cord is passed through the holes and bound around the manuscript to keep the pages in position. A case is also often used for the newly created manuscript. The case is made from the remnants of the leaves and is sewn together with several layers of fabric.

Figure 2.3: A black mixture is used to make the engraved text easy to read

2.2. Khmer Language

Khmer is the official language spoken by Cambodian people. In the next sections, we will talk briefly on how Khmer language has been revolutionized. Different categories of symbols composing Khmer script are also presented.

2.2.1. Revolution of Khmer Language

Throughout Cambodian history, Khmer has been affected significantly by languages such as Sanskrit and Pali under religious influences from Hinduism and Buddhism. It is the earliest recorded written language of the Mon-Khmer family which is the language being used during historical empires of Chenla, Angkor, and presumably even the earlier predecessor state, Funan [12]. According to [13], the history of Khmer language is divided into four periods. The Old Khmer period, which is subdivided into pre-Angkorian (from 600 through 800 AD) and Angkorian (Khmer Empire era from 9th to 13th centuries), is only known from words and phrases in Sanskrit texts of the era. Following the end of Khmer Empire, the Middle Khmer took form under the transitioning period from around the 14th to 18th centuries. During this period, the language underwent a major change in morphology, phonology, and lexicon which is influenced by neighboring countries due to geographical proximity. In the early 20th century, the language once again transitioned through a standardization phase called Khmerization by getting rid of foreign elements, reviving affixation, and using the old Khmer roots (from historical Pali and Sanskrit) to develop new words

for modern ideas. Until today, the language became recognizable as Modern Khmer.

2.2.2. Khmer Script

The Khmer script was one of the earliest writing systems used in Southeast Asia. It derived immediately from the Pallava script of South India. Its alphabet consists of a large number of different types of symbols which can be categorized as follows:

- **Consonants:** there are 35 Khmer consonants. However, Modern Khmer only uses 33 consonants due to two of them becoming obsolete. Every consonant is attached with an inherent vowel and therefore can be a standalone character when being spelled in a word. Each consonant (except one) has a subscript form which is also called "sub-consonant". Most sub-consonants resemble the corresponding consonant counterpart but in a smaller and possibly simplified form although in a few cases, certain consonants and their sub-consonants do not share obvious resemblance. The majority of sub-consonants are written below the main consonant with the exception of several sub-consonants whose parts are elongated from the bottom of the main consonant to either its left or right side.
- **Dependent Vowels:** most Khmer vowel sounds are written using one type of vowels called dependent vowels. Dependent vowels can only be written in combination with a consonant or a consonant cluster (see Figure 2.6) i.e. they cannot stand alone.
- **Independent Vowels:** in contrast to dependent vowels, Khmer independent vowels (also known as "complete vowels") are vowel characters that stand alone i.e. they are not written in combination with other consonants or vowels. The independent vowels are used in only a small number of words.

All Khmer consonants (and their corresponding sub-consonants), dependent vowels, and independent vowels are listed in Appendix A.

In Khmer writing, there are also additional symbols including:

- **Diacritics:** Khmer uses several diacritics to indicate further modifications in pronunciation such as shortening the length of the vowels or to showcase some loanwords from Pali and Sanskrit.
- **Punctuation:** just like most languages, Khmer also uses punctuation to state end of phrases or sentences. In addition to its own punctuation, western-style punctuation marks are quite commonly used in modern Khmer writing.
- **Numeral:** the numerals of the Khmer script, consisting of 10 distinct digits, are also derived from the southern Indian script. Western-style Arabic numerals are also used, but to a lesser extent.

Commonly used diacritics and punctuations as well as all 10 Khmer numerals are also listed in Appendix A.

2.3. Challenges for a DIA System Development

For Khmer palm leaf manuscripts, several major challenges need to be overcome. Those challenges mainly come from the physical conditions as well as the format of palm leaf documents. The complexity of Khmer script also provides many difficulties in solving DIA tasks for these documents.

2.3.1. Conditions and Characteristics of Historical Khmer Palm Leaf Manuscripts

Despite the availability of advanced image capturing methods, photography technology, and scanning equipment, the quality of many palm leaf images is still low due to natural aging. Palm leaf manuscripts are organic in nature and are susceptible to different types of deterioration which leads to physical damage and decay. Some of the most common deteriorating agents are light, insects, constant handling, adverse storage, and climatic factors including variations in relative humidity and temperature. Some degradations and noises caused by these agents include seepage of ink or bleed through, damage or tear around the area of the holes used for binding the document, stain from dirt, and other types of discoloration. Figure 2.4 illustrates these

degradations. The fragility of the aging leaves also presents some difficulties during the digitization process of the leaves. For instance, sometimes leaf manuscripts are curved and cannot be forced flat which results in an uneven illumination in the output image.

Due to their characteristics, palm leaf manuscripts provide challenging layout analysis problems, segmenting palm leaf pages into individual text lines for example. In most Khmer palm leaf manuscripts, the scripters tend to exaggerate their writing by elongating the upper or lower part of a character which makes it go far out of its main line, touch, or overlap with other characters from adjacent lines. Moreover, due to high width-height ratio of the manuscript page, text lines are written very close to each other with very little spacing between them. Because text lines are long, they may also be slanted, curved upward/downward, or fluctuated on account of them being handwritten or from improper digitizing. In addition, strings used to tie palm leaves together to create a book-like binding leave behind holes surrounded by empty areas in the middle of the page producing discontinuity of text lines. Efficient line segmentation approaches on palm leaf manuscript are therefore very important as well as challenging to develop.

Figure 2.4: Several types of deformations and defects found in palm leaf manuscripts

2.3.2. Complexity of Khmer Script

Complexity of Khmer script is also a big challenge. Khmer is recognized by the Guinness World Records[3] to be the language with the longest alphabet. As mentioned in Chapter 2.2.2, certain types of letters have more than one form and/or can be combined with other letters to create more shapes which increase even more the number of symbols in Khmer writing. The abundance of different symbols in Khmer script requires a complex and sophisticated system for those letters to be efficiently recognized and accurately classified.

On account of the large quantity of symbols, many of those symbols are very similar and can be distinguishable by only the appearance of some small strokes or holes and their spatial locations. In old handwritten form, this similarity is even more apparent and sometimes creates an ambiguity between symbols which requires context from neighboring symbols so that those ambiguous symbols can be correctly identified. Some types of symbols contain multiple parts whose shapes are identical or very similar to other characters. Figure 2.5 shows some examples of this ambiguity.

Consonants in Khmer script are used either as individuals or as clusters of multiple letters i.e. a double or triple decker form which is composed of a normal consonant and one or two sub-consonants to merge the sound of those consonants together. Figure 2.6 shows some examples of different combinations of consonant clusters. Vowels and diacritics can be ascenders or descenders or can be placed at either side (right or

Figure 2.5: Example of similarity between certain Khmer characters

[3] http://www.guinnessworldrecords.com/world-records/longest-alphabet

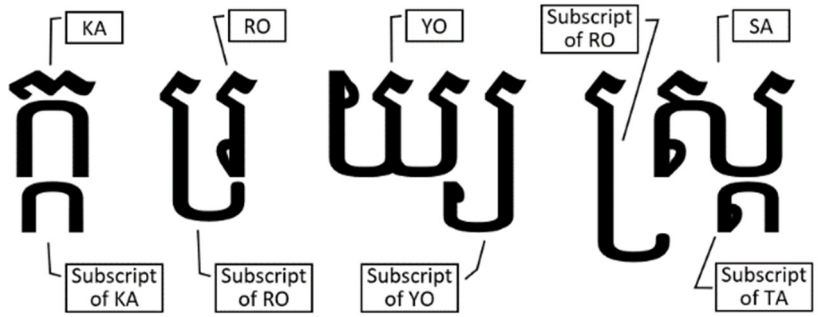

Figure 2.6: Examples of double-decker and triple-decker clusters of Khmer consonants

left), on top, or at the bottom of the main consonant or the cluster of the main consonant. Some letters which consist of multiple parts can even be positioned at different locations simultaneously. Moreover, some ligatures between certain letters produce new shapes which might not be a straightforward combination of the composing letters. Some examples of this type of ligatures are presented in Figure 2.7. In Figure 2.7 (a), a combination between consonant BA and vowel A creates a new shape to represent the word BA-A otherwise their direct combination would be identical to the shape of consonant HA. In Figure 2.7 (b), by attaching vowel E above certain consonants, the vowel replaces the top part of those consonants.

Unicode encoding (U1780-U17FF) has been adopted to represent Khmer symbols (Appendix B). Even though the overall writing direction of a word is left to right, the order of the Unicode codes in the code sequence representing that word does not always follow the writing order of the composing symbols. Also, symbol to code relationship is not always one to one i.e. some symbols can be represented by more than one code, and some codes can represent a combination of symbols. For instance, each subscript of any consonant does not have its own code but is instead represented by a sequence of two codes: a special code “coeng” (U17D2) followed by the code of its corresponding normal consonant. Unlike words in Latin script whose symbols can be identified one by one, to recognize a Khmer word, one must look at the whole writing of the word. This illustrates that the

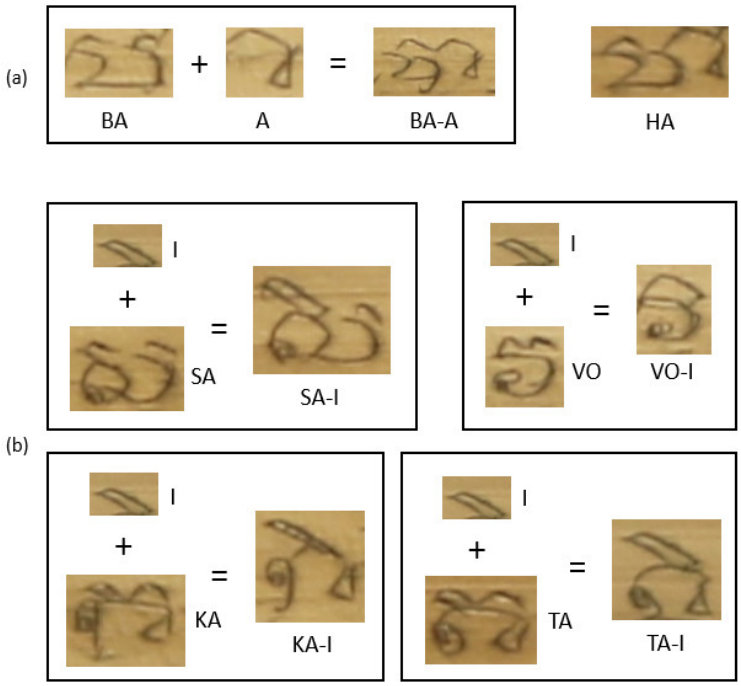

Figure 2.7: Some examples of ligatures between certain letters which form new shapes

spatial information of each symbol composing a word is crucial for the recognition of that word.

Khmer is written from left to right; however, there is no word separation in Khmer writing. Spaces are occasionally used to separate phrases instead of words. We rely on grammatical structures, character combination rules, and sometimes contextual meaning of the sentence in order to identify where the beginning and the ending of a word are. As mentioned earlier, the sequential order of characters composing a word is irregular. For instance, a word must start with a consonant and may be followed by vowels. However, the physical positions of the vowels can either be on the left, on the right, on top, or under the starting consonant. Some particular vowel symbols consist of multiple parts which are placed simultaneously at multiple locations around the consonant. Figure 2.8 illustrates this case. The vowel symbol AEU is composed of two parts which are located on the left and on top of the

main consonant SA. To emphasize even more, we also simulate how an English word could have been written in Khmer style in Figure 2.9.

Figure 2.8: Irregular sequential ordering of symbols in a word. The different order of the position of the characters in the text image (top) and the Unicode sequence of the text (bottom)

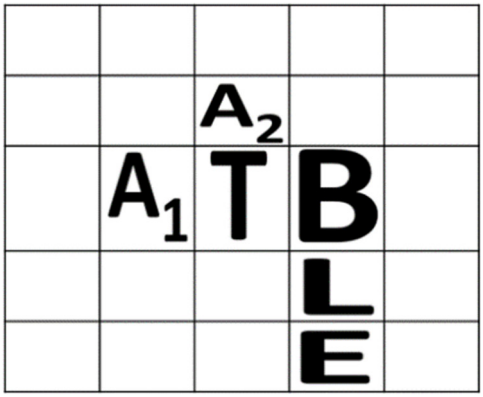

Figure 2.9: Simulation of how the word "TABLE" could have been written in Khmer style (vowel 'A' composed of two parts which are placed on top and on the left side of 'T', consonant 'L' written as a sub-from under 'B', and vowel 'E' placed under 'B' and the sub-f

3. State of the Art: Document Image Analysis

In recent years, research in DIA has received considerable attention. The process of solving DIA tasks does not consist of a single step but often a sequence of steps composing a complete pipeline. For handwritten text recognition problem in particular, the pipeline is divided into two major parts: preprocessing and recognition. The state of the art of each part of the handwritten text recognition scheme which often comprises of different sub-tasks will be discussed in the following sections.

3.1. Preprocessing Tasks

Preprocessing of document images is very crucial and influences greatly the performance of the text recognition stage. In this part, we discuss existing approaches of two preprocess tasks: binarization and line segmentation.

3.1.1. Binarization

Binarization is widely applied as the first preprocessing step in image document analysis [14]. Binarization is a common starting point for document image analysis pipeline. It converts gray image values into a binary representation for background and foreground, or in a more specific definition, text and non-text, which is then fed into further tasks such as text line segmentation and text recognition (OCR). The performance of binarization techniques has a great impact and directly affects the performance of the recognition task [15]. Non optimal binarization methods produce unrecognizable characters with noises [16]. Many binarization methods have been reported. These methods have been tested and evaluated on different types of document collections. Based on the choice of the thresholding value, binarization methods can be generally divided into two types, global binarization and local adaptive binarization [16]. Some surveys and comparative studies of the performance of several binarization methods have been reported [15, 17]. A binarization method that performs well for one

document collection, might not necessarily be applied to another document collection with the same performance [14]. For this reason, there is always a need to perform a comprehensive evaluation of the existing binarization methods for a new document collection with different characteristics, for example, the historical archive documents [17].

3.1.1.1. Global Thresholding

Global thresholding method is one of the simplest techniques and is one of the most conventional approaches for binarization [14, 18]. A single threshold value is calculated from global characteristics of the image. This value should be properly chosen based on a heuristic technique or a statistical measurement to be able to give a promising optimal binarization result [17]. It is widely known that using a global threshold to process a batch of archive images with different illumination and noise variation is not a proper choice. The variation between images on foreground and background colors on low quality document images gives unsatisfactory results. It is difficult to choose one fixed threshold value which is adaptable for all images [17, 19].

Otsu's method is a very popular global binarization technique [14, 18]. Conceptually, Otsu's method tries to find an optimum global threshold on an image by minimizing the weighted sum of variances of the objects and background pixels [14]. Otsu's method is implemented as a standard binarization technique (a built-in Matlab function called graythresh[4]).

3.1.1.2. Local Thresholding

To overcome the weakness of the global binarization technique, many local adaptive binarization techniques were proposed, for example Niblack's method [14, 17, 18, 19, 20], Sauvola's method [14, 17, 18, 19, 20, 21], Wolf's method [19, 20, 22], NICK's method [20], and Rais's method [14]. The threshold value in local adaptive binarization technique is calculated in each smaller local image area, region, or window. Niblack's method proposed a local thresholding computation

[4] https://fr.mathworks.com/help/images/ref/graythresh.html

based on the local mean and local standard deviation of a rectangular local window for each pixel on the image. The rectangular sliding local window covers the neighborhood for each pixel. Using this concept, Niblack's method was reported to outperform many thresholding techniques and gave optimal acceptable results for many document collections [17]. However, there is still a drawback of this method. It was found that Niblack's method works optimally only on the text region, but it is not well suited for large non-text regions on the image. The absence of text on the local areas forces Niblack's method to detect noises as text. The suitable window size should be properly chosen based on the character and stroke size which may vary on each image.

Many other local adaptive binarization techniques were then proposed to improve the performance of the basic Niblack's method. For example, Sauvola's method is a modified version of Niblack's method. Sauvola's method proposes a local binarization technique to deal with light textures, big variation and uneven illuminations. The improvement from Niblack's method is on the use of adaptive contribution of standard deviation in determining local threshold on the gray values of text and non-text pixels. Sauvola's method processes separately the image in $N \times N$ adjacent and non-overlapping blocks.

Wolf's method tries to overcome the problem of Sauvola's method when the gray values of text and non-text pixels are close to each other by normalizing the contrast and the mean of gray values of the image to compute the local threshold. However, a sharp change in the background gray values across the image decreases the performance of Wolf's method. Two other improvements for Niblack's method are Nick's method and Rais's method. Nick's method proposes a threshold computation derived from the basic Niblack's method and Rais's method proposes an optimal size of window for the local binarization.

3.1.1.3. Training Based Binarization

The top two proposed methods in Binarization Challenge for ICFHR 2016 Competition on the Analysis of Handwritten Text in Images of Balinese Palm Leaf Manuscripts [8] are training-based. The best method in this competition employs a Fully Convolutional Network

(FCN). It takes a color sub image as input and outputs the probability that each pixel in the sub image is part of the foreground. The FCN is pre-trained on normal handwritten document images with automatically generated ground truth binarizations using the method of [22]. The FCN is then fine-tuned using DIBCO and HDIBCO competition images and their corresponding ground truth binarizations. Finally, the FCN is fine-tuned again on the provided palm leaf images. For inference, the pixel probabilities of foreground are efficiently predicted for the whole image at once and thresholded at 0.5 to create a binarized output image.

The second-best method uses two neural network classifiers C1 and C2 to classify each pixel as background or not. Two binarized images B1 and B2 are generated in this step. C1 is a rough classifier which tries to detect all the foreground pixels while probably making mistakes for some background pixel. C2 is an accurate classifier which should not classify the background pixel as foreground pixel while probably missing some foreground pixels. Second, they join these two binary images to get the final classification result.

3.1.2. Line Segmentation

Line separation is one of the most important and challenging pre-processing tasks in handwritten text recognition especially for the case of old Khmer handwriting on palm leaf documents. A text line is normally composed by some words which are arranged in such spatial position so that it represents the reading order of all words of the document in the horizontal direction. The vertical position of some text lines gives an important information about the paragraph which represents the layout of the document. Segmentation of document images into physical spatial entities such as text lines, words, and characters is often performed prior to the recognition step of a text recognition system. Segmentation-based text recognition methods need a prior segmentation process of the document image into text line segments and afterward into even smaller units such as word segments or character segments. In this case, extracting properly the text lines in a document enables an easier extraction of smaller size entities of the

document. Consequently, the performance of the recognition system is greatly influenced by the result of the segmentation process.

Even though some of the text line segmentation methods perform sufficiently well for printed documents, segmenting text lines in handwritten documents is significantly more challenging. The difficulties are even more elaborating in historical documents. Some challenges for segmenting text lines from Khmer palm leaf documents are discussed in Chapter 2.3.1. Certain distinct properties of Khmer palm leaf documents can also be noted. As mentioned in Chapter 2.1.2, the manuscript pages are made from dried palm leaves with light yellowish or brown color. Ancient Khmer characters are carved on each page, and a mixture of coal powder and a type of paste is applied on the carving resulting in dark black text. We can therefore be sure that the documents contain dark color foreground text over light color background. Furthermore, despite its form being elaborative and complex, Khmer handwriting is not cursive. Connected components representing individual characters can be extracted from each text line.

Numerous state-of-the-art methods of line segmentation have been proposed. However, many of them operate only on binary images. A variety of projection profile techniques are used [23, 24, 25, 26]. The goal of this type of methods is to extract estimated medial or seam positions of text lines based on the peaks/valleys of the histogram profiles projected on vertical axis of the document page. Another genre of line segmentation approaches tries to find optimal paths which pass through text line seams [27, 28, 29, 30]. Tracers following the white-most and black-most trails are used in [27] in order to shred the image into text line areas. In [28], an improved Viterbi algorithm based on Hidden Markov Model generates all possible paths, and a filtering process is applied afterwards to remove invalid paths leaving behind only the optimal ones. To select the ideal paths, the approach in [29] computes an energy map which accumulates energy from left to right for each path while a traveling cost function of the path is instead calculated in [30]. Other methods of text line segmentation working on binary images include smearing/blurring [31], Hough transform [32], and water flow [33]. For the approaches mentioned above, a good initial

binarization process is required. However, common degradations and noises on aging palm leaves such as seepage of ink or bleed through, tears, stains from dirt and moisture, and other types of damage render this task impossible to produce an acceptable result. It is therefore necessary that a new binarization-free scheme is developed.

Some methods which are independent on the binarization process have also been proposed. In [34], projection profile is applied directly on a grayscale document. The profile is computed by summing the pixel values from each row. Different skew angles can be detected and estimated. However, one of the drawbacks from the method is that seams separating text lines are straight along the direction of the detected skew. This is not suitable for documents with little spacing between adjacent lines whose text components may touch or be too close to each other that the seams are not able to separate them correctly. Another technique transforms the document page into a set of interest points which can be grouped into clusters to form text lines based on their local orientation [35]. The approach of [36] makes use of seam carving technique to compute separating seams between text lines by constraining the optimization procedure inside a defined region. Similarly, in [37], areas for segmentation path are selected, and the multi-stage graph search algorithm is applied to find the shortest nonlinear path in each area.

Due to specific characteristics of Khmer handwriting, text line extraction from Khmer palm leaf manuscripts still remains an open problem. By taking into account some properties of this type of document such as its non-cursive writing style, a performance improvement over state-of-the-art methods is expected. The new proposed scheme should manage to deal with challenges including detecting skew/fluctuation of text lines and especially the discontinuity of text lines which is difficult to solve for many of the existing approaches.

3.2. Character and Text Recognition

Character or text recognition is the final task in the DIA pipeline. This task analyses an input image (normally already segmented), extracts

useful information from it, and returns a corresponding text as output. The extracted information is a group of values called features which are reduced from the initial set of raw data in the input image. Traditionally, feature extraction techniques are often non-training based i.e. the methods are manually engineered for certain specific tasks. Those handcrafted features are then passed to a classifier (for example, Nearest Neighbor or SVM) to output the predicted class label of the character image or the text transcription of the text image. Recently, due to the availability of high-performance computing resources as well as the accessibility to labeled data of document images, training-based approaches become more and more popular since features can be extracted automatically and possibly more efficiently. Next, well known handcrafted feature extraction methods followed by training-based approaches such as artificial neural networks and deep learning will be discussed.

3.2.1. Handcrafted Feature Extraction

Numerous feature extraction methods for character and text recognition have been proposed in the literature [38, 39, 40, 41]. Each method is designed to deal with certain specific problems. Reviews on feature extraction techniques were also reported [42, 43, 44].

Some traditional approaches include projection histogram [45], crossing [46], outer contour profile [47], and Kirsch directional edge [48]. The projection histogram technique counts the total number of pixels composing the foreground texts or characters row-wise (horizontal histogram) or column-wise (vertical histogram) while the crossing approach counts the number of pixel transitions from background to foreground or vice versa. The profile method computes the distance from the contour of the object to be recognized to the boundary of the input image. For Kirsch direction edge method, feature vectors are extracted from the segmented regions of the binary edge image constructed by computing the edge strength from neighboring pixels in four directions (horizontal, vertical, and the two diagonals). Other popular traditional techniques consist of Fourier transform [49] and Moment invariants [50]. Zoning [51] is also often used. This

technique divides the image into smaller partitioned zones. The division can be performed to create one-dimensional sequences of row-wise, column-wise, or diagonal-wise slices as well as multi-dimensional zones such as grid-like cells, circular zones, or radial zones. Any feature extraction method can then be applied independently on each zone. For instance, the Celled Projection (CP) technique [52] extracts the projection histogram features from each of the sub-divided zones.

For palm leaf manuscripts in particular, [53] conducts a study on some popular features and evaluates them on the dataset of digitized palm leaf documents. In this study, 10 feature extraction methods, some of them being traditional approaches described above, are investigated. After evaluating the performance of those individual feature extraction methods, the Histogram of Gradient (HoG) features as directional gradient based features [38], the Neighborhood Pixels Weights (NPW) [54], and the Kirsch Directional Edges combined with Zoning are found to give very promising results. A new feature extraction method by applying NPW on Kirsch edge images is also proposed, and the concatenation of the NPW-Kirsch output with the two other features (HoG and Zoning) is then chosen as the final feature vectors.

3.2.2. Neural Networks and Deep Learning

3.2.2.1. Neural Network Basics

Artificial Neural Networks (ANNs) were originally developed as a mathematical tool to be used to model information processing capabilities of human brain [55]. The architecture of basic ANNs consists of processing units or nodes which are connected to each other by weighted connections i.e. the outputs of some neurons might become inputs to other neurons. Many variations of the basic ANNs have been proposed over the past years [56]. One of the widely used ANN structure is feed forward in which instead of indeterminate groups of connected neurons, the ANN models are structured into a layer-wise organization (the models are organized as distinct layers of neurons). There are always one first layer called input layer and one last layer called output layer while the intermediate layers between the first and the last layer are called hidden layers. The most common layer type for

regular ANNs is the fully connected (FC) layer. In a FC layer, neurons between two adjacent layers are fully pairwise connected, but neurons from the same layer share no connection with each other. Mathematically, a neuron k can be described by the following equations:

$$u_k = \sum_{i=1}^{m} w_{ki} x_i \quad (3.1)$$

$$y_k = f(u_k + b_k) \quad (3.2)$$

where $x_1, x_2, \ldots, x_m$ are the inputs, $w_{k1}, w_{k2}, \ldots, w_{km}$ are the weights of neuron k, b_k is the bias, and f is the activation function which represents the frequency of the activation of neuron k (or simply put, it decides whether the neuron should be fired or not). In other words, each neuron in the FC layer performs a dot product with the input and its weights, adds the bias, and applies the non-linearity to output a fired signal to neurons in the next layer.

Activation Function

Activation functions take a single value and performs a certain fixed mathematical operation on it [57]. Commonly used activation functions are:

- Sigmoid has its mathematical form $f(x) = 1/(1 + e^{-x})$ which squashes the input real-value number into range between 0 and 1. This type of non-linearity has two drawbacks: (1) sigmoids saturate and kill gradients giving the vanishing gradient problem and (2) sigmoid outputs are always positive and not zero-centered so the gradient on the weights during back propagation become either all positive or all negative which could introduce undesirable dynamics in the gradient updates for the weights.
- Tanh $f(x) = \tanh(x)$ squashes the input number to the range between -1 and 1. It is often preferred to sigmoid in practice since it eliminates one of the drawbacks mentioned earlier (tanh is zero-centered).
- ReLU (short for Rectified Linear Unit) computes the formula $f(x) = \max(0, x)$. The activation simply threshold the input

number at zero. Compared to sigmoid and tanh, due to its linear and non-linear form, the weight update converges much faster (no saturation). The ReLU operation is also much less expensive. There is also a variant of ReLU called Leaky ReLU [58] which instead of the function being zero when $x < 0$, a small negative slope α is used to compute the output value: $f(x) = \alpha x$ if $x < 0$ otherwise $f(x) = x$.

Figure 3.1 illustrates these activation functions.

Cost Function

Cost functions (sometimes also referred to as objective function, loss or error) are used to estimate how the networks perform. A cost function is a measure of how wrong the network is in terms of its ability to estimate the relationship between the input output pairs. This is typically expressed as the difference or the distance between the output values predicted by the network and the real values (also known as the

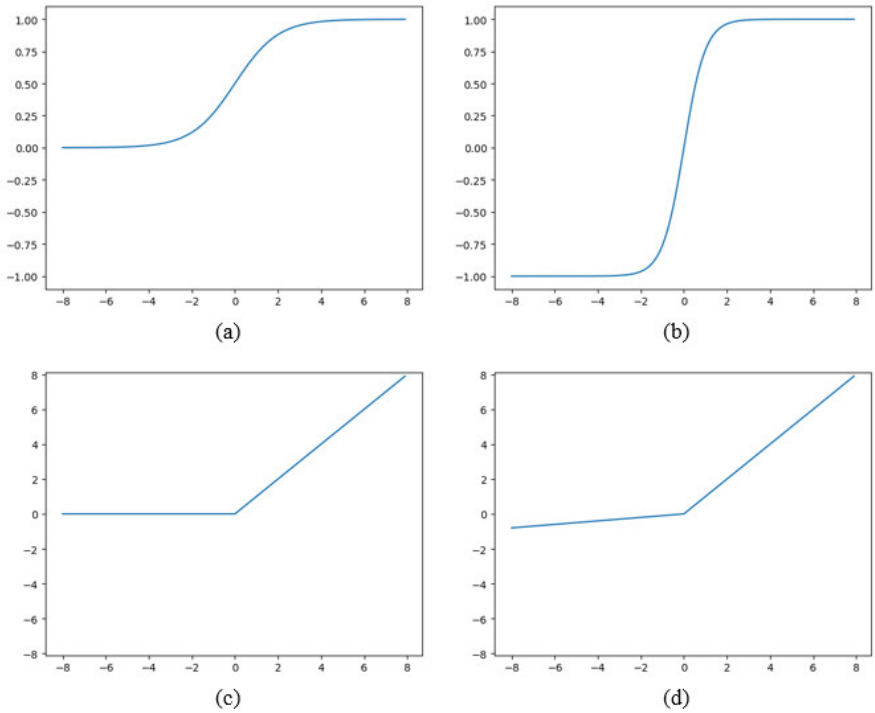

Figure 3.1: (a) Sigmoid, (b) Tanh, (c) ReLU, (d) Leaky ReLU

ground truth). By minimizing the cost function i.e. finding the optimal parameters (normally weights and biases), we can make sure that the network performance (its predicting ability) is as good the ground truth.

Gradient descent is one of the efficient algorithms that attempt to find a local or global minimum of a function. It allows the network to compute the gradient or the direction that the network should take from parameters of consecutive layers in order to reduce losses (difference between the predicted output and the ground truth). Gradient descent therefore enables the learning process to make corrective updates to the learned estimates that move the network toward an optimal combination of parameters.

Backpropagation

The forward propagation refers to when we use a feedforward neural network to process an input and produce an output where the input contains initial information that flows forward trough hidden units at each layer. The backpropagation algorithm computes information from the cost or objective function and then lets it flow backward through the network in order to determine the gradient for the weights at each layer [59, 60]. By using the chain rule, backpropagation enables us to simultaneously compute all the partial derivatives using just one forward pass followed by one backward pass through the network. For neural networks, their layer-like architectures can be seen as a computational graph. For any input value fed into the network, layer by layer during the backward pass, the gradient of each parameter that we passed through during the forward pass is computed in a reversed manner.

3.2.2.2. Convolutional Neural Networks

Convolutional Neural Networks (CNNs), a variant of the feed forward network, are a specialized kind of network for processing data with a grid-like topology [61, 60]. Similar to ordinary feed forward ANNs, CNNs are made up of neurons with learnable weights and biases. What makes CNNs different is the use of mathematical operation called “convolution” (hence the name). Convolution can be considered as a

process where we take a small matrix of numbers (called kernel or filter) and pass it over a normally bigger matrix and transform it based on the values from the filter. The output values are calculated according to the following formula:

$$(g * h)[m, n] = \sum_i \sum_j h[i, j] g[m - i, n - j] \qquad (3.3)$$

where the input matrix is denoted by g and the kernel by h. The indexes of rows and columns of the output matrix are denoted by m and n respectively.

CNNs are therefore ANNs that use convolution in place of matrix multiplication in at least one of their layers (such layers are called convolutional layers). For applications related to digital images, the convolutional layers take an input volume of dimension $height \times width \times depth$ (for example an image with 3 color channels has a depth of 3) and return an output volume of feature maps. The parameters in the convolutional layers can be seen as a set of learnable filters. Each filter is often small spatially along the width and height dimension but extends through the full depth of the whole input volume. During the forward pass, each filter is shifted or slid across the width and height of the input volume to apply the convolution operation i.e. the dot products between the entries of the filter and the region of the input volume that we slide the filter on. Equation 3.3 is therefore extended as

$$(g * h)[m, n] = \sum_i \sum_j \sum_k h[i, j, k] g[m - i, n - j, k] \qquad (3.4)$$

where h and g now are 3-dimensional (height, width, and depth). The network will intuitively learn filters that activate when patterns such as edges, corners, or some type of visual features are found. For each filter, after being slid over the entire region of the input volume, a 2-dimensional feature map is produced. According to the number of filters used in the layer, the feature maps can be stacked together, and an output volume of feature maps is obtained.

Pooling layer is commonly used in-between successive convolutional layers to progressively reduce the spatial size of the representation

output volume of feature maps. This consequently also reduces the number of parameters and the computation cost in the network. The most common form of pooling layer is the maxpooling with filters of size 2×2 and with a stride of 2. It uses MAX operation which selects the maximum value from each 2×2 region of each individual slice along the depth dimension of the input volume. After passing through this maxpooling layer, the input volume will be downsampled by a factor of 2 along the width and height dimension keeping only a quarter of the activations of the feature map volume (the depth dimension of the output however normally remains unchanged).

Figure 3.2 presents an overall architecture of a simple CNN. The network takes as input a one-channel image with dimension 28×28. The first convolutional layer uses 32 filters of size 5×5, and it is followed by a maxpooling layer. The second convolutional layer uses 64 filters of the same size as in the first convolutional layer, and its output is again passed through a maxpooling layer to reduce its size. The output volume is then transformed into a vector to be fed to the first fully connected layer and then the second one to produce the final class probabilities.

In this dissertation, the application of CNNs on document images is focused. CNNs have been tremendously successful in practical applications on digital image analysis since the architectures allow us to encode certain properties of the input images. Some eminent deep CNN architectures including AlexNet [62], VGGNet [63], and ResNet [64] are considered to be state-of-the-art approaches for image classification problems.

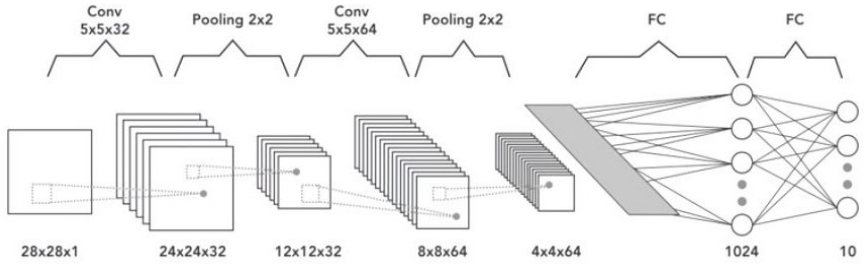

Figure 3.2: Architecture of a simple CNN

3.2.2.3. Recurrent Neural Networks

Recurrent neural networks (RNNs) are a family of neural networks for processing sequential data [59]. In contrast to feedforward networks, RNNs contain loops that return output activations back into the network. Basically, an RNN remembers the past and its decisions are influenced by what it has learned from the past. In other words, a simple feedforward ANN takes a fixed size input vector and transforms it into a fixed size output vector. Such a network becomes "recurrent" when the transformation is performed repeatedly instead on each element in a sequence of input vectors to produce a sequence of output vectors. Figure 3.3 is an example of a simple RNN architecture with one hidden layer. Let $(x_1, x_2, \dots, x_t, \dots, x_N)$ be a sequence of input vectors which are fed into the RNN in order to produce an output sequence $(y_1, y_2, \dots, y_t, \dots, y_N)$ where N is the number of elements in both sequence. At each time step t, the RNN has an internal or hidden state h_t which can be computed by the input vector in the current time step x_t and the hidden state in the previous time step h_{t-1} as following:

$$h_t = f(W_x x_t + W_h h_{t-1} + b) \qquad (3.5)$$

where W_x, W_h, b are weights and bias respectively and f is an activation function. It should also be noted that the parameters W_x, W_h, and b are shared for every computation at each time step t. The hidden state h_t can be used to further compute the corresponding output y_t.

Backpropagation Through Time

Backpropagation Through Time (or BPTT) happens when the backpropagation algorithm is applied to train RNNs. As mentioned earlier, a RNN processes a sequence of inputs, and one input is shown to the network at each timestep to produce one output which is then fed back to the network to be used in the computation for the next timestep. This behavior causes the network to contain loops (rolled version) as illustrated in Figure 3.3.a. The network, however, can be unrolled i.e. each timestep of the unrolled network (Figure 3.3.b) can be considered as an additional layer given that the output from the previous timestep is taken as input to the layer. As such, similar to feedforward networks,

we determine derivatives through each timestep of the unrolled network (hence the term "through time"), compute gradients, and finally reroll the network to update weights. The main difference here is that unlike in the feedforward networks where the errors are normally computed only at the end of the network (the output layer), the errors in RNNs are accumulated across each timestep taking into account also the order dependence of the input sequence.

Long Short-Term Memory

RNNs seek to establish connections between a final output and events many time steps before. It is therefore difficult to know the importance of those remote inputs since the information flows through the neural networks by passing through many stages of multiplications. Any quantity multiplied frequently by an amount greater than one can become large quickly while multiplying by a quantity less than one saturates to zero. In RNNs, due to the relation between time steps and multiplications, the weight gradients (derivatives) are susceptible to exploding and vanishing. Exploding gradients happen when the weight gradients become saturated on the high end (they become too large). This can be solved by truncating or squashing the values of the

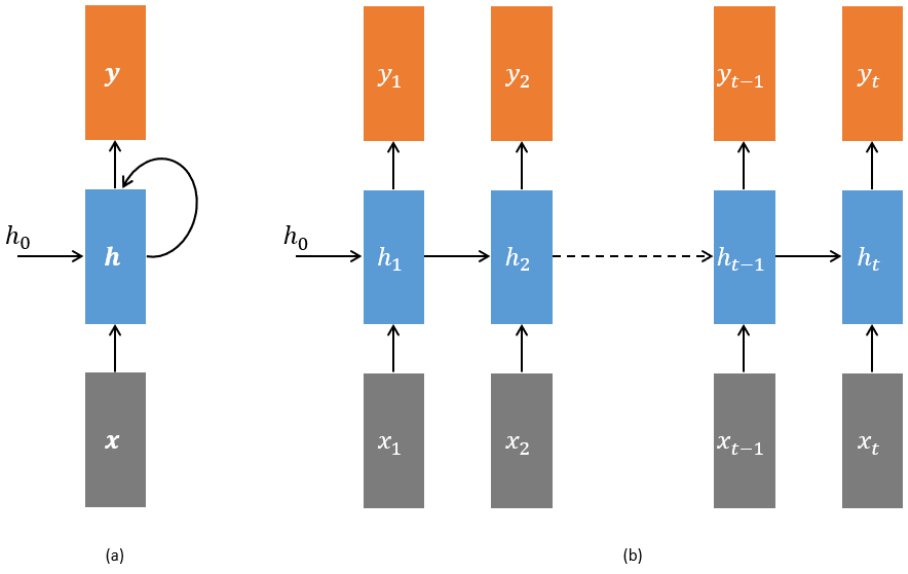

Figure 3.3: Architecture of a simple RNN (a) rolled, (b) unrolled

gradients. Vanishing gradients (when the gradients are too small or very close to zeros) are however harder to solve.

Long Short-Term Memory RNNs or LSTMs in short [65, 66] are designed to overcome the vanishing gradient problem and allow them to retain information for longer periods (time steps) compared to traditional RNNs. Figure 3.4 illustrates the architecture of an LSTM cell. As seen in Figure 3.4, LSTMs use gated cells to store information in addition to the regular flow of the RNN. Due to these cells, the network is able to manipulate the information in many ways, including storing information in the cells and reading from them. Each cell in the LSTM unit is capable of making decisions regarding the information and can execute these decisions by opening or closing the gates.

The three major parts of an LSTM cell are: (1) the forget gate which eliminates information that becomes unnecessary, (2) the input gate which is responsible for adding new information to the cells, and (3) the output gate which selects and outputs the necessary information corresponding to each time step. The formula for each gate at each time step t is computed as follows:

$$\Gamma_f = \sigma(W_f[x_t; h_{t-1}] + b_f) \quad (3.6)$$

$$\Gamma_i = \sigma(W_i[x_t; h_{t-1}] + b_i) \quad (3.7)$$

$$\Gamma_o = \sigma(W_o[x_t; h_{t-1}] + b_o) \quad (3.8)$$

where Γ_f, Γ_i, and Γ_o represent the forget gate, the input gate, and the output gate respectively. Each gate uses its own weights and biases and is computed as a function of the hidden state from the previous time step h_{t-1} and the current input vector x_t.

In addition to the hidden state h_t, LSTM cells contain another state called "cell state" denoted by C_t. This cell state maintains information from previous time steps. As mentioned earlier, the LSTM cell is able to add new useful information or remove unnecessary one. This modification is performed on the cell state via the forget gate (to remove information) and the input gate (to add information). The new information to be added is denoted by $\widetilde{C_t}$ and is computed by

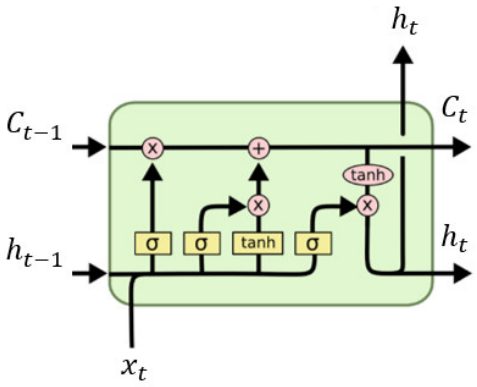

Figure 3.4: Architecture of an LSTM cell

$$\widetilde{C_t} = tanh\,(W_C[x_t; h_{t-1}] + b_C) \qquad (3.9)$$

The new cell state C_t is modified by calculating how much information should be kept (or removed) from the previous cell state C_{t-1} and how much new information should be added from $\widetilde{C_t}$. The modification is as follows

$$C_t = \Gamma_f \odot C_{t-1} + \Gamma_i \odot \widetilde{C_t} \qquad (3.10)$$

where $\odot$ denotes element-wise multiplication. Finally, the hidden state h_t is computed by determining how much information should be output from the cell state C_t using the output gate. Before going through the output gate, C_t is first squashed by tanh so that the values are between -1 and 1. The hidden state is therefore calculated by

$$h_t = \Gamma_o \odot tanh\,(C_t) \qquad (3.11)$$

Numerous variants of LSTM RNNs have been proposed using a slightly different version from the basic LSTM architecture described above. One popular LSTM variant is adding special connections called "peepholes" which allows the computations of the three gates to take into account also the information from the cell state [67]. Another variation which also gains popularity in recent years is the Gated Recurrent Unit or GRU [68]. It combines the cell state and the hidden state and also merges the forget gate and the input gate into a single gate called the "update" gate among other changes. Other notable variants

include the Depth Gated LSTM proposed by [69] and the Clockwork RNNs [70] which tackles long-term dependencies using a completely different approach. Some comparison studies of popular variants of LSTM have been conducted [71, 72]. It is found that some variants work better than the other on certain tasks; however, their overall performances are somehow similar. For the sake of simplicity, in this dissertation, the basic LSTM will be focused.

Bi-directional RNNs

Bidirectional RNNs (BRNN) introduced by [73] are an extension to typical RNNs that can enhance the performance of the model on sequence classification problems. The idea behind BRNNs involves using two recurrent layers (instead of just one) to process the input sequence twice. The first layer handles the input sequence as it is while the second layer takes a reversed copy of the input sequence which adds additional context to the network. In other words, BRNNs allow us at a point in time to take information from both earlier and later in the sequence. Figure 3.5 gives an example of a BRNN.

At each time step t, two hidden states can be obtained: one from the forward direction ($\vec{h}_t$) and another one from the backward direction ($\overleftarrow{h}_t$). To take into account information from both directions, the two hidden states can be merged together before being passed to the next layer. The final hidden state at each time step h_t can be computed by

$$h_t = \delta(\vec{h}_t, \overleftarrow{h}_t) \qquad (3.12)$$

where δ is a merge function. The most common options for δ are

- Sum: the hidden states are added together element-wise.

$$h_t = \vec{h}_t + \overleftarrow{h}_t \qquad (3.13)$$

- Average: the average of the hidden states is computed.

$$h_t = \frac{\vec{h}_t + \overleftarrow{h}_t}{2} \qquad (3.14)$$

- Multiplication: the hidden states are multiplied together element-wise.

$$h_t = \vec{h}_t \odot \overleftarrow{h}_t \tag{3.15}$$

- Concatenation: the hidden states are concatenated together doubling the dimension of h_t.

$$h_t = [\vec{h}_t; \overleftarrow{h}_t] \tag{3.16}$$

Multi-dimensional RNNs

The standard RNN architectures are explicitly one dimensional, meaning the input sequence needs to be pre-processed to one dimensional as well. Data such as digital images are two dimensional in nature, and in order to be fed to an RNN, the input images are preferably transformed into sequence of slices along either width (column wise) or height (row wise) dimension. The RNN therefore is not able to exploit the full multi-dimensional structure of the images. Multi-dimensional RNNs (MDRNNs) have been proposed by [74] to solve this problem. The basic idea behind MDRNNs is to use as many

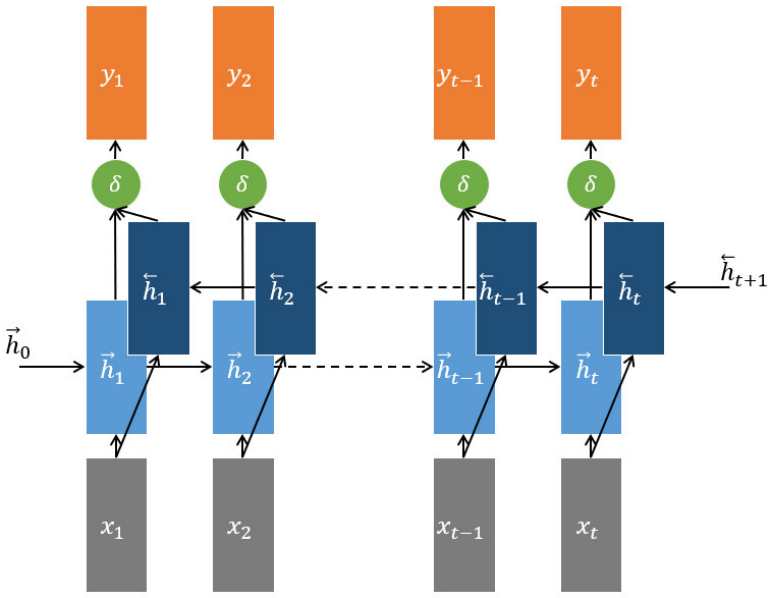

Figure 3.5: Architecture of a simple Bidirectional-RNN

recurrent connections as there are dimensions in the data to replace the single recurrent connection found in standard one-dimensional RNNs. Even though MDRNNs are designed to perform on multi-dimensional data, in this dissertation, only the application on images which are two dimensional will be discussed.

LSTMs, a variant of RNNs, can also be extended to multi-dimensional (MDLSTMs) [74]. Instead of a single hidden state from the previous time step like in the conventional one-dimensional LSTM, MDLSTM, 2D-LSTM in particular, makes use of two states each from both the vertical and horizontal axes. Denote $h_{t-1}^{(1)}$ and $h_{t-1}^{(2)}$ as the previous hidden states, $C_{t-1}^{(1)}$ and $C_{t-1}^{(2)}$ the previous memory (cell) states along the two axes, and x_t the current input, we compute the current hidden state h_t and the current memory state C_t of the 2D-LSTM as follows:

$$\Gamma_i = \sigma\left(W_i\left[x_t; h_{t-1}^{(1)}; h_{t-1}^{(2)}\right] + b_i\right) \tag{3.17}$$

$$\Gamma_f^{(1)} = \sigma\left(W_f^{(1)}\left[x_t; h_{t-1}^{(1)}; h_{t-1}^{(2)}\right] + b_f^{(1)}\right) \tag{3.18}$$

$$\Gamma_f^{(2)} = \sigma(W_f^{(2)}\left[x_t; h_{t-1}^{(1)}; h_{t-1}^{(2)}\right] + b_f^{(2)}) \tag{3.19}$$

$$\Gamma_o = \sigma\left(W_o\left[x_t; h_{t-1}^{(1)}; h_{t-1}^{(2)}\right] + b_o\right) \tag{3.20}$$

$$\tilde{C}_t = tanh\left(W_C\left[x_t; h_{t-1}^{(1)}; h_{t-1}^{(2)}\right] + b_C\right) \tag{3.21}$$

$$C_t = \Gamma_i \odot \tilde{C}_t + \Gamma_f^{(1)} \odot C_{t-1}^{(1)} + \Gamma_f^{(2)} \odot C_{t-1}^{(2)} \tag{3.22}$$

$$h_t = \Gamma_o \odot tanh\,(C_t) \tag{3.23}$$

where W's and b's are weights and biases, and Γ_i, Γ_f, and Γ_o are respectively input, forget, and output gates of the LSTM. Since it is two-dimensional, there are two forget gates corresponding to each of the two previous memory states and computed with different sets of parameters.

Multi-directional RNNs

The concept of bi-direction in one-dimensional RNNs can also be applied to MDRNNs. For two-dimensional data such as images, we

would prefer the network to have access to surrounding contexts from all directions. One example of directional order in two-dimensional sequence of a digital image is that we set the origin of the two axes (let's denote the horizonal axis z_1 and the vertical axis z_2) at the top left corner of the image, and $x_{i,j} < x_{i',j'}$ ($x_{i,j}$ is processed by the network before $x_{i',j'}$ in the sequence) if $i < i'$ or $j < j'$ in case $i = i'$ where $x_{i,j}$ represents an element in the sequence at the i^{th} row and the j^{th} column from the origin. Similarly, we can define the other three directional orders by setting the origin of the two axes at the top right, at the bottom left, and at the bottom right of the image respectively as shown in Figure 3.6. By using these four directional orders, the network produces independently four hidden states corresponding to each direction which can be merged together by summing, multiplying, averaging, or concatenating as in Equation 3.12.

3.2.2.4. Neural Networks for Handwriting Recognition

CNNs have been successfully used for solving isolated character recognition problems, for example they are considered to be the base line approaches for the popular but simple handwritten digit dataset (MNIST) [75, 76]. As mentioned earlier, one of the main benefits of CNNs is that the input images are fed into the networks as raw pixel values to be processed to extract useful automatic features instead of the handcrafted ones.

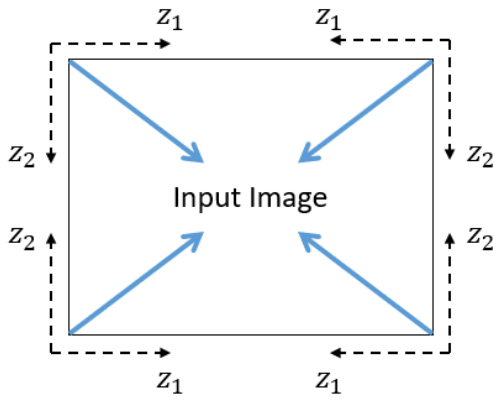

Figure 3.6: Axes representing the 4 possible directions in 2D-RNN

For text recognition, due to texts being sequences of smaller units such as characters or glyphs, the problem becomes a sequence learning task. Over the last few years, end-to-end handwritten text recognition models using RNNs have started to outperform earlier approaches [77] such as those based on Hidden Markov models [78, 79]. LSTM and its extended version such as Bi-directional LSTM as well as its multi-dimensional variant have been used. The current state-of-the-art in many handwritten text recognition tasks additionally integrates CNNs as an improved low-level feature extraction module prior to the recurrent layers. Some recent works have shown great success in using solely RNN modules or the combination with CNNs for text recognition tasks of different languages. Some examples include Latin texts which are the most primitive ones [80, 81, 82, 83], Chinese and Japanese scripts which show numerous challenges such as the large number of character classes and vocabularies [84, 85, 86, 87], and also Arabic handwritten documents with the cursive nature of the writing and the visual similarities of the characters [88, 89]. However, the main challenge in processing sequential data is to find appropriate alignment information that matches elements in the input sequence to those of the output sequence. Currently, the two major directions to solve this problem are (1) the use of Connectionist Temporal Classification (CTC) introduced by [90] and (2) the sequence to sequence (Seq2Seq) technique.

CTC is a type of objective function which can be attached at the end of the recurrent modules. CTC-based architectures remove the need to forcefully align the input stream with character prediction location. This provides a benefit of not needing properly segmented labeled data. Nevertheless, these architectures are subject to inherent limitations like strict monotonic input-output alignment i.e. one-to-one input-output pair. This technique is one dimensional in nature and works efficiently well for scripts with one directional writing style (for example, left to right or top to bottom), in other words no more than one character is at the same horizonal or vertical position. However, for scripts with a more complex writing style such as Khmer, character annotation and alignment information might still be required to produce a more accurate recognition result.

On the other hand, the main concept of Seq2Seq architectures is that they follow the encoder-decoder framework. The models consist of two main parts i.e. they decouple the decoding from the feature extraction module. First, an encoder reads and builds a feature representation of the input sequence, then a decoder emits the output sequence one token at a time. Usually an attention mechanism is employed by the decoder to gather context information and search for relevant parts of the encoded features.

In Chapter 7, following the Seq2Seq scheme, we propose a model which takes advantage of both the convolutional module and the multi-dimensional recurrent module specifically developed to recognize texts on Khmer palm leaf manuscripts. The proposed model also incorporates the annotated spatial alignment information of each character or glyph in the text image. Moreover, due to its depth, these types of deep architectures mentioned above normally require significantly large amount of data (for example tens of millions of samples) to train and tend to overfit on smaller datasets. Some optimization is also taken into account of designing the proposed model. Data augmentation technique is also employed in order to extend existing data.

4. Digital Image Corpus and Ground Truth Dataset

This chapter presents the collection of digital corpus of Khmer palm leaf manuscripts, the construction of ground truth tool and data, and the introduction to SleukRith Set, the first Khmer palm leaf manuscript dataset. We also talk briefly about datasets constructed on palm leaf manuscript from Indonesia which will be used in addition to the datasets from SleukRith Set for the experimental studies in the next chapter.

4.1. Digital Image Corpus

4.1.1. Existing Digital Images of Khmer Palm Leaf Manuscripts

In Cambodia, existing digital data of palm leaf manuscripts can be found in various libraries and institutions. Table 4.1 shows the number of digital image collections (a collection here refers to one complete set of palm leaf documents) available in those establishments. Sample images are shown in Figure 4.1.

École Française d'Extrême-Orient (EFEO): the online database of Khmer manuscripts is the result of the work conducted by the École Française d'Extrême-Orient (EFEO-FEMC) research team since 1990, aiming to provide a comprehensive inventory and photographic collection of Cambodia's manuscripts. The website is accessible to public and is home to hundreds of collections of palm leaf manuscripts. The digital images were captured from microfilms hence their low quality.

Buddhist Institute (BI): the Buddhist Institute was an initiative of King Sisovat in 1921, when he inaugurated the royal library, Khemra Bannalai which subsequently changed its name to Preah Raj Bannalai in 1925. Then in 1930, King Monivong established the Buddhist Institute. The responsibilities of the Institute are not only research on Cambodian literature, language and Buddhism, but also publication and education. Only one collection containing 96 pages is available from this institute. The digitization method of this collection is unknown.

Phnom Penh National Library (NL): the National Library of Cambodia was inaugurated by the French colonial administration. Thereafter it was successively managed by French staff until the appointment of the first Khmer Director in 1951. After independence in 1954 there was a steady growth in Cambodian publishing, which was reflected in the increased number of Khmer language books in the National Library. Closed down during the Khmer Rouge era, the National Library was used for several years as accommodation by members of the Pol Pot regime, who destroyed many of the books. Since 1980 the National Library has been re-established with the assistance of various overseas governments and agencies. Today the National Library of Cambodia holds hundred thousands of books in various languages (Khmer, French, English, and German). There is also a large collection of palm leaf manuscripts. Some collections of the manuscripts were digitized recently by a Khmer manuscript conservation and research group. We obtained in total 35 collections of already digitized manuscripts from this establishment.

Table 4.1: Number of digital Image collections available in various establishments

N°	Establishment	Digitization Method	Nb. Of Collections[5]
1	École Française d'Extrême-Orient (EFEO)	Nikon F3	937
2	Buddhist Institute (BI)	Unknown	1
3	National Library (NL)	Canon 750D	35

4.1.2. Digitization Campaign

We also conducted our own digitization campaign in order to capture and collect palm leaf manuscript images found in Buddhist temples in different locations throughout Cambodia. A standard digitization procedure and a proper set up have been developed. Due to the fact that palm leaf manuscripts are fragile, and certain scripts are not allowed to be moved from the place where they are stored, digitizing using a

[5] Each collection consists of in average several dozens of palm leaf pages

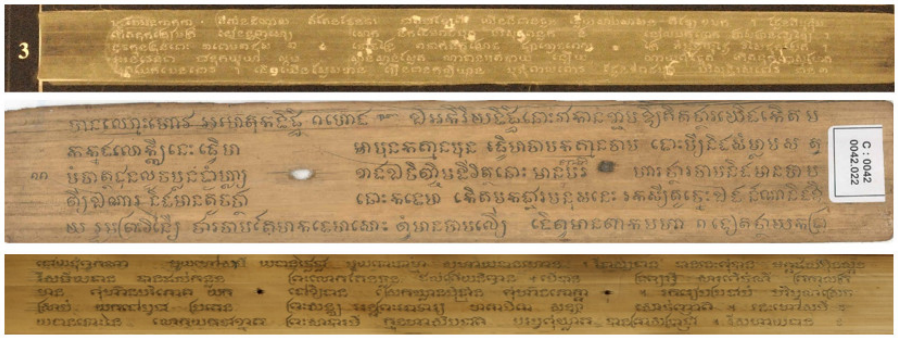

Figure 4.1: Samples of digitized images from top to bottom: EFEO, BI, and NL

scanner is not viable, so a portable option was needed. In order to capture the scripts, a Canon EOS 5DS professional camera was used. The camera settings are as follows: F-stop: f/4, shutter speed: 1/10 second, ISO: 100, focal length: 45 mm, distance to object: 65 cm, and auto focus: on. To support the camera to be able to shoot downward, we use a Manfrotto 055XPro3 tripod with its fluid head. To avoid irregular lighting condition and to adapt to our semi indoor/outdoor capturing location, the camera is covered over by a black cloth. Additional rechargeable led lights (led 715) are attached to each of the tripod legs. Figure 4.2 illustrates this set up.

Our campaign has been conducted in three locations in Cambodia: Phnom Penh, Kandal, and Siem Reap. We have collected and digitized manuscripts found mostly in Buddhist temples (pagodas). A summary of the collection from our digitization campaign is listed in Table 4.2. Some sample images are shown in Figure 4.3.

Table 4.2: Collection of digitized palm leaf manuscripts from our digitization campaign

N°	**Location**	**Nb. Of Collections**	**Nb. Of Pages**
1	Tuol Tom Poung, Phnom Penh	2	54
2	Tek Vil Pagoda, Kandal	2	98
3	Bo Pagoda, Siem Reap	9	59
Total		**13**	**211**

Figure 4.2: A set up for our digitization campaign

Figure 4.3: Sample images of our digitization campaign (from top to bottom: Phnom Penh, Kandal, and Siem Reap)

4.2. SleukRith Set

4.2.1. Description of SleukRith Set

SleukRith set is a collection of three types of annotated data: isolated characters, words, and lines. The annotation is made on 657 pages of Khmer palm leaf manuscript randomly selected from different sources. A summary of selected pages and their sources are shown in Table 4.3. The majority of the images are chosen from the recently digitized

manuscripts at the National Library and from our digitization campaign. Due to their low quality, the dataset contains much fewer pages from the collections of EFEO and the Buddhist Institute.

For annotating the three types of data, a tool with an easy-to-use user interface has been developed. The tool is implemented in Java hence its multi-platform portability. Since annotating a large amount of data can be quite exhausting, most interactions between the user and the tool are performed using only left and right clicks of mouse buttons. A keyboard input is required when giving labels to the data, modifying data, or deleting data.

The annotation process was accomplished with the help from volunteer students from the department of Computer Science at the Institute of Technology of Cambodia (ITC) and the National Institute of Posts, Telecommunications, and ICT (NIPTICT). Each participant played the role of a ground truther and was assigned a set of palm leaf document images identified by number codes. Using their common knowledge of Khmer language, the ground truthers were asked to annotate each of their assigned pages according to the following steps: segment and label all characters, group segmented characters into words, and finally assign each segmented character to a line it belongs to. After the initial annotation stage done by the students, a final validation and correction iteration has been performed verifying that the data is consistent and without errors.

Table 4.3: Collection of palm leaf manuscript images from different sources composing SleukRith Set

Nº	**Source**	**Nb. Of Pages**
1	National Library	427
2	EFEO	26
3	Buddhist Institute	15
4	Our Digitization Campaign	189
Total		**657**

4.2.2. Isolated Glyphs

The individual or isolated character dataset is the most important data type in SleukRith Set since its information is used to produce the other types of data. In order to segment and annotate a manuscript page into small image patches representing each individual character, a polygon boundary enclosing the character needs to be drawn manually. The ground truther is required to dot out vertex of the polygon one by one until a proper boundary is formed (see Figure 4.4). The ground truther is then prompted to input the correct Unicode or Unicode sequence as label for that character.

A problem in annotating a character occurs when it is composed of multiple parts. In this situation, each part of the character is segmented separately and is labeled with the original Unicode of the character followed by a number representing that part. A different situation is when multiple characters are merged together and form a new shape. In this case, the shape is then annotated as a whole and is given the label which is a sequence of the Unicode of the characters comprising it. Examples illustrating these cases are shown in Figure 4.5.

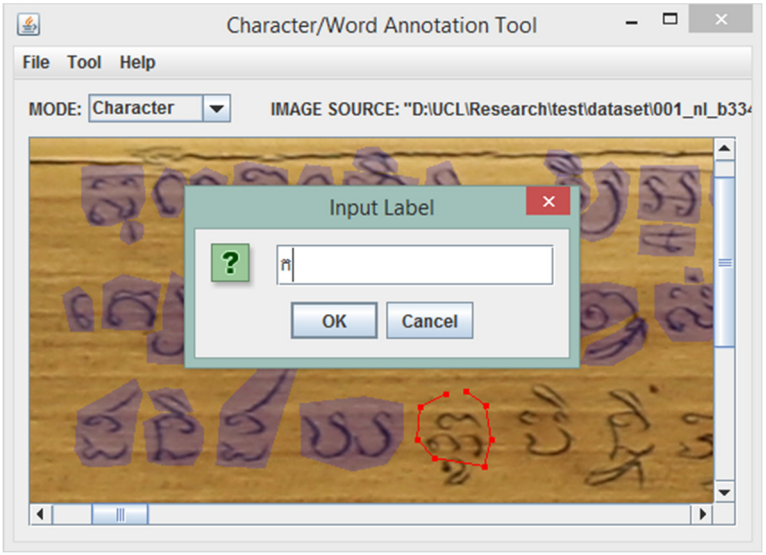

Figure 4.4: Annotation of individual character dataset

Certain writers exaggerate their writing by elongating the ending stroke of the characters. Also, some characters are written in a way that they encircle other characters. When cropped into a rectangular area, the image patch of such elaborate character does not only contain the character itself but also parts or the entirety of other characters. To solve this issue, by using the polygon boundary as a mask, an inpainting technique [91] can be applied in order to eliminate the unwanted area in the image patch. In Figure 4.6, the image patch of character SUBYO is inpainted resulting in a new clean image.

4.2.3. Words

After all characters in the page are manually annotated, they can be combined together into words. To form a word, the character components of that word are selected one by one (see Figure 4.8). The selection order is also important since Khmer Unicode sequence does not follow the left to right position order of the characters but instead respects a consonant-first-vowel-second basis. Figure 4.7 shows an example illustrating this phenomenon. In the example, the word is composed of five characters, and the correct sequence is PHO-SUBLO-

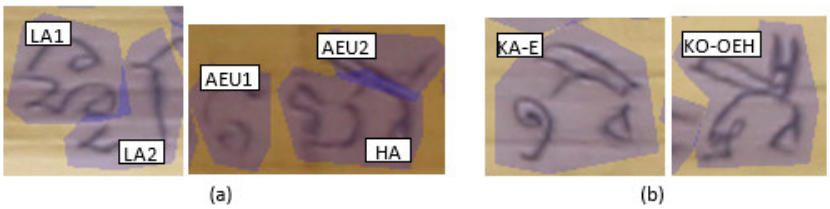

Figure 4.5: Examples of (a) characters containing multiple parts and (b) merged shapes

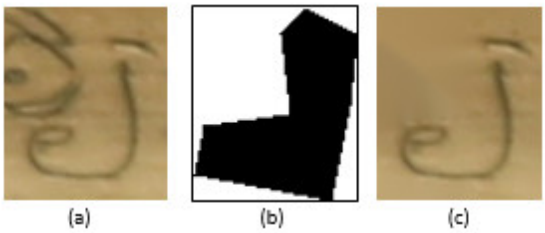

Figure 4.6: Application of inpainting technique on a character image patch (a) input image, (b) inpainting mask using polygon boundary, (c) result

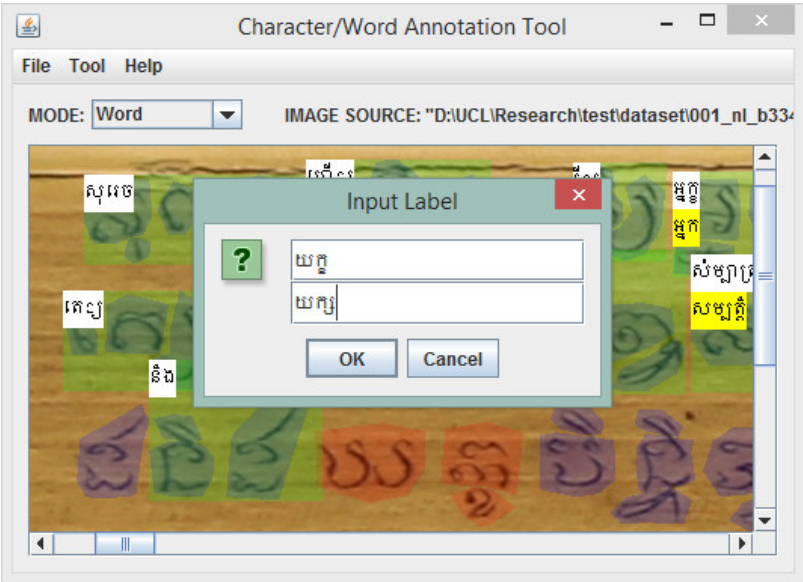

Figure 4.8: Annotation of word dataset

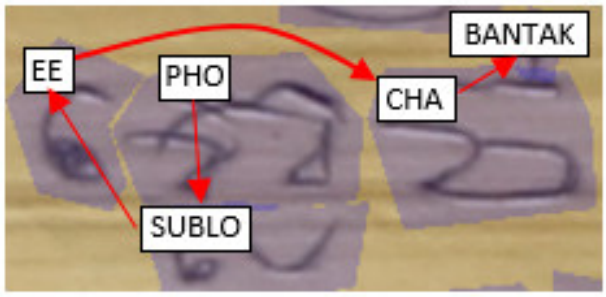

Figure 4.7: Order sequence of characters in a word

EE-CHA-BANTAK. Even though the vowel EE is at the left-most position of the word, it is placed third in the Unicode sequence after the consonant PHO and the sub-consonant SUBLO.

The ground truther is then again prompted to input a Unicode sequence representing the label of the formed word. By default, the word label is generated by putting together the labels of the characters which are the components of that word. The second label should also be provided by the ground truther when either the current word spelling is found to be erroneous or when an equivalent word from the modern Khmer language has a different spelling.

4.2.4. Sub-syllables

One of the main difficulties in creating useful data from palm leaf manuscripts is that the number of digitized pages is still limited. Current machine learning approaches require much more data to train, and the number of annotated words in the SleukRith Set is still limited and insufficient. Moreover, some manuscripts are written in Pali which, even though it uses Khmer alphabet, is a completely different language from Khmer. It requires the knowledge of the language to identify the word separation, and therefore word annotation and recognition cannot be achieved easily for those manuscripts.

To solve both of these problems, we introduce a new type of data added to the SleukRith Set called "sub-syllables". A sub-syllable refers to a group of glyphs which satisfies the following criteria:

1) A sub-syllable must be a cluster of glyphs belonging to the same text line and containing at least one main consonant or be a standalone digit, a punctuation, or an independent vowel
2) In the case where it is a cluster of glyphs, it must not begin with any irregular character such as an dependent vowels, a sub-consonant, or a diacritic
3) Each sub-syllable must represent the smallest possible combination of glyphs which satisfies criteria 1) and 2). A new label is also needed to be specified for the created sub-syllable.

Adjacent sub-syllables can be joined together to form a group representing a synthetic word. Let n-SG denote a group of sub-syllables where n is the number of sub-syllables composing it. A text line containing N sub-syllables might therefore be able to produce up to $(N - n + 1)$ n-SGs. Two neighboring sub-syllables are not grouped together if the distance between them is greater than a predefined threshold. This is to take into account blank gaps which often appear between phrases in the manuscripts. Figure 4.9 illustrates a comparison between word separation and sub-syllable separation. As shown, eight sub-syllables can be extracted from the text. We can group therefore those sub-syllables into n-SGs. For $n = 2$, the following set of 2-SGs can be generated: {(1, 2), (2, 3), (3, 4), (5, 6), (6, 7), (7, 8)} i.e. we can

Figure 4.9: (a) Sample Khmer text, (b) Word separation, (c) Sub-syllable separation

obtain six synthetic words instead of only four real words. Note that the sub-syllable 4 and 5 cannot be grouped together since there is a large gap between them.

4.2.5. Lines and their Transcriptions

Similar to word and sub-syllable annotation, annotated characters may be grouped into lines. To efficiently achieve this, using the annotation tool, the ground truther uses left click and drag over annotated characters belonging to the same line. He is then asked to create a new line from the selected characters or add them to existing lines (see Figure 4.10). Each annotated glyph is given an ID corresponding to the line which it belongs to. The area of a text line can therefore be constructed by performing a union operation on all its glyph polygon boundaries. Due to the irregularity of how certain glyphs are positioned in the text sequence, the text transcription cannot be generated by simply using a left-to-right sequential concatenation of the labels of all glyphs in the line. Another benefit of annotated sub-syllables is that they can be used to generate line transcriptions. Similar to annotated words, a rectangle boundary of a sub-syllable is built using the polygon boundaries of the component glyphs. We then sort all sub-syllables in a line by their horizontal positions and produce the final line transcription

by concatenating the label of each sub-syllable according to this sorted order.

4.2.6. Annotation File Format

After all steps in the annotation scheme are complete, an xml file containing all information of the four types of data of the annotation can be exported for each manuscript page. It also serves as a temporary save file to store incomplete progress of the annotation. The xml file is divided into five sections (see Figure 4.11). The first section contains meta information related to the document page. The information includes the 3-digit code, the document name, the document source, and the resolution of the document image. The next four sections represent respectively the four types of annotation. The section under the tag name "CharAnno" is dedicated to the annotation at the character level. This section block contains child blocks. Each child block represents an annotated character, information about the coordinates of its polygon boundary and additional attributes including character id, its label, and the id of the line which the character belongs to. The next section under the tag name "WordAnno" describes the annotation at the word level. Since a word is a combination of characters, only the IDs of the

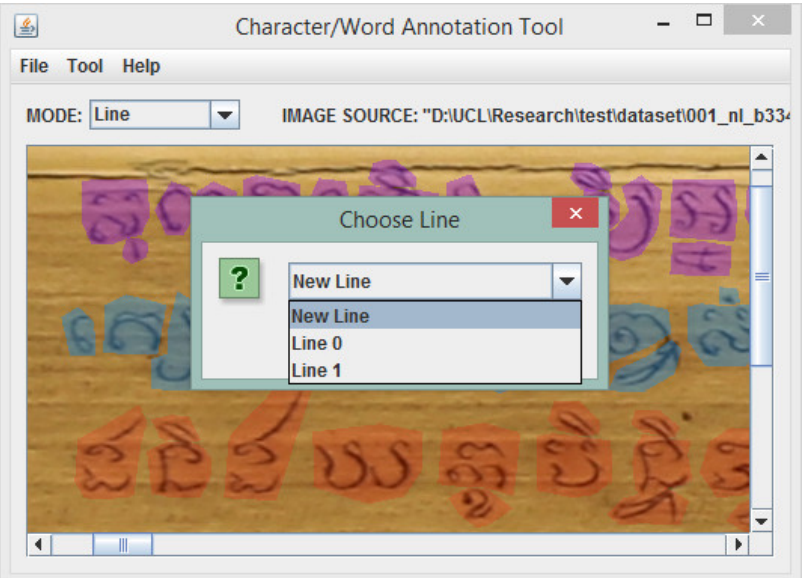

Figure 4.10: Construction of line segmentation ground truth

```
<Annotation>
  <MetaData code="008"
            imName="008_nl_a161_304_01.jpg"
            location="nl"
            imHeight="468"
            imWidth="5470" />
  <CharAnno>
    ...
    <Char id="5" label="ꦲ" lineid="0">
      <Poly x="718" y="65"/>
      <Poly x="726" y="117"/>
      <Poly x="777" y="112"/>
      <Poly x="775" y="69"/>
    </Char>
    ...
  </CharAnno>
```

```
<SubSylAnno>
  ...
  <SubSyl id="7" label="ꦲꦼ">
    <CharInSubSyl id="13"/>
    <CharInSubSyl id="15"/>
    <CharInSubSyl id="14"/>
  </SubSyl>
  ...
</SubSylAnno>
<WordAnno>
  ...
  <Word id="7" label="ꦲꦼꦤ꧀" label2="ꦲꦤ꧀ꦠꦂ">
    <CharInWord id="30"/>
    <CharInWord id="31"/>
    <CharInWord id="33"/>
    <CharInWord id="34"/>
    <CharInWord id="32"/>
  </Word>
  ...
</WordAnno>
```

```
  <LineAnno>
    <Line id="0"/>
    <Line id="1"/>
    <Line id="2"/>
    <Line id="3"/>
    <Line id="4"/>
  </LineAnno>
</Annotaion>
```

Figure 4.11: Sample of an xml file storing annotation information of a manuscript page

annotated characters defined in the previous section are stored along with the id information of the annotated word and its two labels. The sub-syllable annotation is stored under the section with tag name "SubSylAnno". Similar to the word annotation, each sub-syllable is given an ID and is represented by a block containing child blocks storing the IDs of all glyphs composing the sub-syllable. Only a single label is noted for each sub-syllable. The last section of the xml file (under tag name "LineAnno") holds information linked to the line annotation. This section stores a number of child blocks equivalent to the number of text lines in the document page.

From this xml file, image patches representing characters and words (or groups of sub-syllables) can be generated (see Figure 4.12 and Figure 4.13). Table 4.4 shows the current statistics of the SleukRith Set.

Table 4.4 Summary of the statistics of the SleukRith Set

Nº	**Data**	**Quantity**
1	Document pages	657
2	Annotated glyphs	302,191
3	Character classes	225
4	Annotated sub-syllables	157,754
5	Unique sub-syllables	3,603
6	Annotated words	73,660

Nº	Data	Quantity
7	Unique words	9,301
8	Text lines	3,247

4.3. Additional Datasets of Palm Leaf Manuscripts from Indonesia

In Indonesia, palm leaves were also historically used as writing supports in manuscripts from Indonesian archipelago. The leaves of sugar palm (Borassus Flabellifer) are known as lontar. Although the official language of Indonesia, Bahasa Indonesia, is written in the Latin script,

Figure 4.12: Samples of annotated character patch images

Figure 4.13: Samples of annotated word patch images

Indonesia has many local and traditional scripts, most of which are ultimately derived from Brahmi. In this dissertation, we focus on two Indonesian scripts: Balinese and Sundanese.

4.3.1. Balinese Manuscripts

In Bali, palm leaf manuscripts were written in Balinese script in Balinese language, in the ancient literary texts composed in the old Javanese language of Kawi and Sanskrit. Balinese language is a Malayo-Polynesian language spoken by about more than 3 million people mainly in Bali, Indonesia. Balinese language is the native language of the people of Bali, known locally as Basa Bali. The alphabet and numeral of Balinese script is composed of more or less 100 character classes including consonants, vowels, and some other special compound characters.

Apart from the collection at the museums (Museum Gedong Kertya Singaraja and Museum Bali Denpasar), it was estimated that there are more than 50,000 lontar collections which are owned by private families. In order to obtain a large variety of manuscript images, the sample images have been collected from 23 different collections (contents), which come from 5 different locations (regions): 2 museums and 3 private families. It consists of randomly selected 10 collections from Museum Gedong Kertya (City of Singaraja, Regency of Buleleng, North Bali), 4 collections from manuscript collections of Museum Bali (City of Denpasar, South Bali), 7 collections from the private families situated in the Village of Jagaraga (Regency of Buleleng), and 2 others collections from the private families in the Village of Susut (Regency of Bangli) and the Village of Rendang (Regency of Karangasem) [92]. Sample images of digitized Balinese palm leaf manuscripts are shown in Figure 4.14.

4.3.2. Sundanese Manuscripts

The collection of Sundanese palm leaf manuscripts (Figure 4.15) comes from Situs Kabuyutan Ciburuy, Garut, West Java, Indonesia. The Kabuyutan Ciburuy is a cultural complex heritage from Prabu Siliwangi and Prabu Kian Santang, The King and the son of the Padjadjaran

Figure 4.14: Balinese palm leaf manuscripts

Figure 4.15: Sundanese palm leaf manuscript

kingdom. The cultural complex consists of six buildings. One of them is Bale Padaleuman which is used to store the Sundanese palm leaf manuscripts. The oldest Sundanese palm leaf manuscript in Situs Kabuyutan Ciburuy came from the 15th century. In Bale Padaleuman, there are 27 collections of Sundanese manuscripts. Each collection contains 15 to 30 pages, with the dimension of 25-45 cm in length x 10-15 cm in width [93]. The Sundanese palm leaf manuscripts were written in the ancient Sundanese language and script. The characters consist of numeral characters, vocal characters, basic characters, punctuations, diacritics, and many special compound characters.

5. Preprocessing

A complete DIA system for Khmer palm leaf manuscripts is composed of two main modules: preprocessing and text recognition. In this chapter, we present experimental studies on the preprocessing part of the DIA system. Two sub-tasks will be covered in the preprocessing step: a benchmarking and comparison study of binarization approaches from the literature which are discussed in Chapter 5.1 followed by a new proposed approach of binary-free line segmentation introduced in Chapter 5.2 along with an experimental evaluation with some base-line approaches.

5.1. Binarization

In this section, we compare several alternative binarization algorithms for palm leaf manuscripts. We test and evaluat some well-known standard binarization methods, and some binarization methods that are promising experimentally for historical archive documents, not specifically for images of palm leaf manuscripts. We also test the binarization methods from DIBCO competition [94, 95] for example the Howe's method [96] and the ones from the binarization challenge of ICFHR competition [8]. The evaluation of the binarization approaches is conducted on three palm leaf manuscript datasets: Khmer and two scripts from Indonesia (Balinese and Sudanese).

5.1.1. Datasets

The palm leaf manuscript datasets for binarization task are presented in Table 5.1. For Khmer dataset, the binary ground truths are created from the digitized EFEO images (see Chapter 4.1.1). When necessary, a local thresholding method [21] is applied, and noises caused by isolated pixels are then removed using median filter. The results are corrected with the help of a photo editing software. The binarized document is superimposed on the original image, and strokes are traced manually using a stylus with pressure sensitive tip to maintain the variation of stroke width of each character in the manuscript. An example of the binary ground truths of Khmer manuscripts is shown in Figure 5.3. For the manuscripts from Bali, the binarized ground truth images have been

created with a semi-automatic scheme [92, 97] (Figure 5.3) while the binarized ground truth images for Sundanese manuscripts [93] have been manually generated by using PixLabeler [98] (Figure 5.3). The training set required for training-based approached is provided only for the Balinese dataset.

Table 5.1: Palm leaf manuscript datasets for binarization task

Manuscripts	**Train (pages)**	**Test (pages)**	**Ground Truth (pages)**	**Dataset**
Khmer	-	46	46	Extracted from EFEO
Balinese	50	50	100	Extracted from AMADI_LontarSet
Sundanese	-	61	61	Extracted from Sundanese Dataset

5.1.2. Evaluation Method

Following the evaluation method from the DIBCO 2009 contest [94] and the ICFHR 2016 competition [8], three metrics of binarization evaluation are used: F-Measure (FM), Peak SNR (PSNR), and Negative Rate Metric (NRM).

F-Measure (FM) is computed from Recall and Precision as following:

$$FM = \frac{2.Recall.Precision}{Recall+Precision} \quad (5.1)$$

$$Recall = \frac{TP}{FN+TP} \quad (5.2)$$

$$Precision = \frac{TP}{FP+TP} \quad (5.3)$$

where TP is a true positive which occurs when the image pixel is labeled as foreground as in the ground truth. FP is a false positive representing when the image pixel is labeled as foreground, but the ground truth is labeled as background, and FN is a false negative which represents when the image pixel is labeled as background, but the

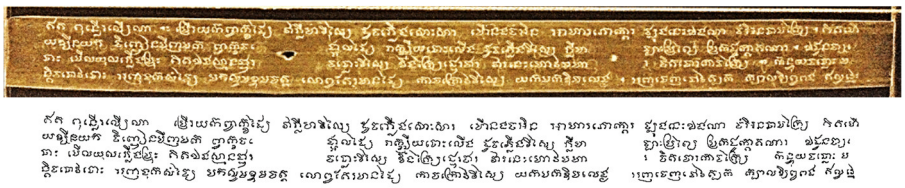

Figure 5.3: Khmer manuscript with binarized ground truth image

Figure 5.3: Balinese manuscript with binarized ground truth image

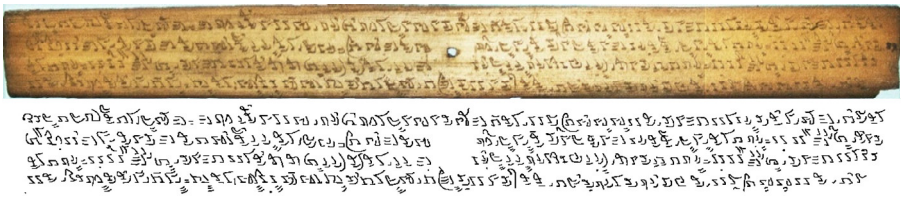

Figure 5.3: Sundanese manuscript with binarized ground truth image

ground truth is labeled as foreground. A higher F-measure indicates a better match.

Peak SNR (PSNR) is calculated from Mean Square Error (MSE):

$$MSE = \sum_{x=1}^{M} \sum_{y=1}^{N} \frac{(I_1(x,y) - I_2(x,y))^2}{M.N} \quad (5.4)$$

$$PSNR = 10.log_{10}\left(\frac{C^2}{MSE}\right) \quad (5.5)$$

where C is defined as 1, the difference between the foreground and the background colors in the case of binary image. A higher PSNR indicates a better match.

Negative Rate Metric (NRM) is computed from the negative rate of false negative (NR_{FN}) and the negative rate of false positive (NR_{FP}).

$$NR_{FN} = \frac{FN}{FN+TP} \quad (5.6)$$

$$NR_{FP} = \frac{FP}{FP+TN} \quad (5.7)$$

$$NRM = \frac{NR_{FN}+NR_{FR}}{2} \quad (5.8)$$

where TN is the true negative which occurs when both the image pixel and ground truth are labeled as background. A lower NRM indicates a better match.

5.1.3. Experiments and Results

The experimental results for the binarization task are presented in Table 5.2. These results show that the performances of all methods on each dataset are still quite low. Most of the methods achieve only less than 50% FM score. It means that binarization on palm leaf manuscripts is still an open challenge. The different parameter values for the local adaptive binarization methods gives a significant improvement in the performance of those methods, but it is still unsatisfactory. In these experiments, ICFHR G1 method was evaluated for Khmer and Sundanese dataset by using the pre-trained Balinese training set weighted model. Based on these experiments, ICFHR G1 method gives the highest FM score for Khmer manuscripts (Figure 5.4), ICFHR G2 gives the highest FM score for Balinese manuscripts (Figure 5.5), and Niblack's method gives the highest FM score for Sundanese manuscripts (Figure 5.6). However, according to the observation on the output results, many broken and unrecognizable characters/glyphs and noises are still visually seen in the binary images.

Table 5.2: Experimental results for the binarization task

Methods	Parameter	Manuscripts	FM (%)	NRM	PSNR (%)
OtsuGray	Otsu from gray image using Matlab graythresh[6]	Khmer	23.92	0.3130	7.38
		Balinese	18.98	0.3988	5.01
		Sundanese	23.70	0.3266	9.99
OtsuRed		Khmer	21.15	0.3371	5.90

[6] https://fr.mathworks.com/help/images/ref/graythresh.html

	Otsu from red image channel using Matlab graythresh	Balinese	29.20	0.3001	10.94
		Sundanese	21.25	0.3864	12.60
Sauvola	window = 50, k = 0.5, R = 128	Khmer	44.73	0.2685	26.06
		Balinese	13.20	0.4623	27.69
		Sundanese	6.19	0.4799	24.78
Sauvola2	window = 50, k = 0.2, R = 128	Khmer	47.55	0.1557	21.96
		Balinese	40.18	0.2745	25.09
		Sundanese	43.04	0.2996	23.65
Sauvola3	window = 50, k = 0.0, R = 128	Khmer	30.55	0.1900	12.78
		Balinese	35.38	0.1658	17.05
		Sundanese	40.29	0.1814	16.25
Niblack	window = 50, k = -0.2	Khmer	38.01	0.1608	16.84
		Balinese	41.55	0.1757	21.24
		Sundanese	**46.79**	**0.1950**	**20.31**
Niblack2	window = 50, k = 0.0	Khmer	30.55	0.1900	12.78
		Balinese	35.38	0.1658	17.05
		Sundanese	40.29	0.1814	16.25
Nick	window = 50, k= -0.2	Khmer	51.25	0.1760	24.51
		Balinese	37.85	0.3283	27.59
		Sundanese	29.59	0.3904	24.26
Rais	window = 50	Khmer	31.59	0.1879	13.52
		Balinese	34.46	0.1710	16.84
		Sundanese	40.65	0.1770	16.35
Wolf	window = 50, k = 0.5	Khmer	46.78	0.2373	25.19
		Balinese	27.94	0.3929	27.16
		Sundanese	42.40	0.2991	23.61
Howe1		Khmer	40.20	0.2806	25.59
		Balinese	44.70	0.2676	28.35

	Default values[7]	Sundanese	45.90	0.2351	21.90
Howe2	Default values	Khmer	32.35	0.2940	25.96
		Balinese	40.55	0.2739	28.02
		Sundanese	35.35	0.2748	22.36
Howe3	Default values	Khmer	30.71	0.3820	26.36
		Balinese	42.15	0.3049	28.38
		Sundanese	25.77	0.3503	23.66
Howe4	Default values	Khmer	36.48	0.2805	25.83
		Balinese	45.73	0.2730	28.60
		Sundanese	38.98	0.2811	22.83
ICFHR G1	See [8]	Khmer	**52.65**	**0.2505**	**28.16**
		Balinese	63.32	0.1500	31.37
		Sundanese	38.95	0.3290	24.15
ICFHR G2	See [8]	Khmer	-	-	-
		Balinese	**68.76**	**0.1300**	**33.39**
		Sundanese	-	-	-
ICFHR G3	See [8]	Khmer	-	-	-
		Balinese	52.20	0.1800	26.92
		Sundanese	-	-	-
ICFHR G4	See [8]	Khmer	-	-	-
		Balinese	58.57	0.1700	29.98
		Sundanese	-	-	-

5.2. Text Line Segmentation

Text line segmentation is one of the most crucial pre-processing steps in the DIA pipeline. In ancient documents, palm leaf manuscripts in particular, a variety of deformations caused by aging produce noises which make the binarization process very challenging as mentioned in

[7] http://www.cs.smith.edu/~nhowe/research/code/

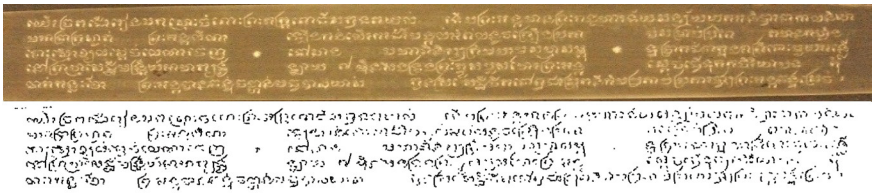

Figure 5.4: Binarization of Khmer manuscript with ICFHR G1 method

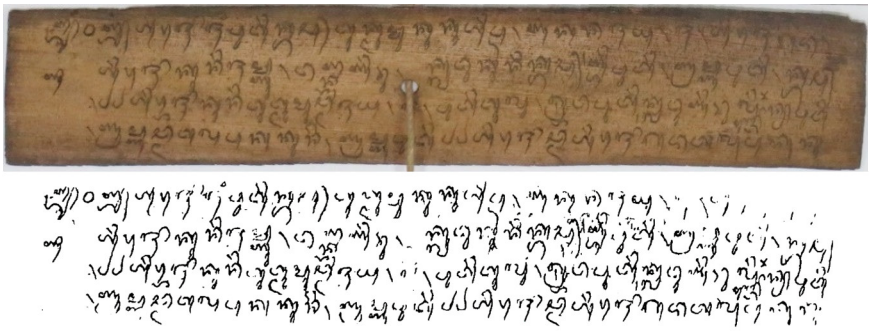

Figure 5.5: Binarization of Balinese manuscript with ICFHR G2 method

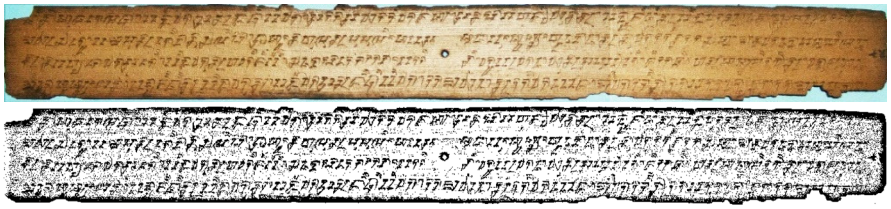

Figure 5.6: Binarization of Sundanese manuscript with Niblack's method

the previous section. Moreover, due to the irregular layout such as skewness and fluctuation of text lines, segmenting an ancient manuscript page into separate lines still remains an open problem to solve. An efficient approach should therefore be binary-free and be able to deal with the complex layout of the palm leaf manuscripts. In this section, we propose a novel line segmentation scheme for grayscale images of Khmer palm leaf documents. First, connected components are extracted from the document page. The number and medial positions of text lines are estimated using a modified piece-wise projection profile technique. Those positions are then modified adaptively according to the curvature of the actual text lines. Finally, a path finding approach is used to separate touching components and also

to mark the boundary of the text lines. An overview pipeline of the proposed text line segmentation approach is presented in Figure 5.7. The input data to the system is a grayscale image of a palm leaf manuscript page. The details of each step of the pipeline are described next. Two experiments are conducted. The first experiment is performed on a small subset of the SleukRith while the second experiment expands to a larger subset in addition to the datasets constructed from the Balinese and Sundanese manuscripts. In both experiments, we also compare the robustness of the proposed approach with existing methods from the literature.

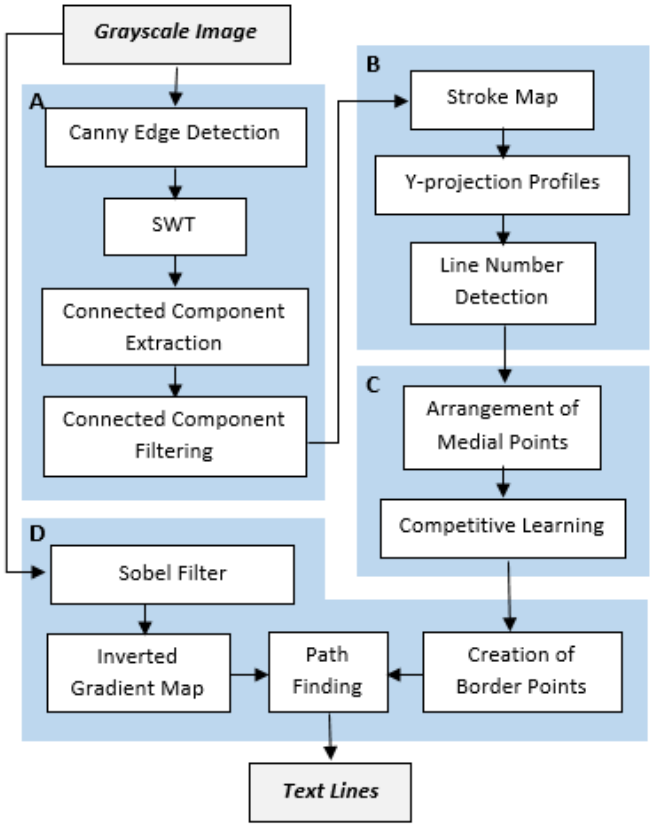

Figure 5.7: Overview of the proposed line segmentation pipeline

5.2.1. Description

5.2.1.1. Foreground Text Detection and Extraction (A)

In order to extract text components from a document page, edges in the image are first calculated using Canny edge detection method [99]. A text localization approach called Stroke Width Transform (SWT) [100] is then applied. SWT is a local image operator which computes per pixel the width of the most likely stroke containing the pixel. In other words, for each pixel in the image, we determine the most relevant stroke that the pixel belongs to. The stroke width value is then given to that pixel. A new feature map of strokes representing foreground texts can be computed by grouping connected pixels whose stroke width values are similar.

The approach creates a new feature map storing the width values of the most likely strokes containing each pixel. We then group together adjacent pixels with similar stroke width into characters (connected components). The average stroke width SW_{avg} of the whole stroke map is also computed. Unwanted noise components such as big patches, long horizontal or vertical strokes, and small dots are removed using the following rules:

- The mean of stroke width values of all pixels in the connected component must be less than $2.SW_{avg}$ to prevent including in the stroke map any undesirable connect component that contains too many large strokes. Small strokes representing text often have consistent width.
- The length-width and width-length ratio of the component must be less than 3. This rule eliminates connected components which are either too wide or too thin i.e. they do not resemble Khmer glyphs.
- The length or width of the component must be greater than $3.SW_{avg}$. Following this rule, noises including small dots and long thin strips are removed.

After all noise components are filtered out, we obtain a new stroke map I_{CC} consisting of all stroke pixels as black foreground over a white

background. The median height H_{median} of all connected components is also calculated. This process is illustrated in Figure 5.8.

5.2.1.2. *Line Number and Location Estimation (B)*

The main goal of this step is to determine the number of text lines in the manuscript page. First, X-projection profile is built from the map to find the starting and ending positions of the text region. The discovered region is divided into N_C vertical columns whose width is empirically set to $W_C = 10.H_{median}$. We then construct a Y-projection profile histogram for each column, and we smooth them twice using moving average filters with widow size of $⅓H_{median}$ and $½H_{median}$ respectively. We consider the local maxima or peaks of the smoothed histogram of each column to be the medial line positions in that column. The variance value of each peak is computed, and spurious peaks with very small value of variance compared to the average variance values of all peaks in their same column are removed. To further filter out incorrect peaks, we also remove peaks too close to each other. Suppose in a column, n peaks are detected. P_i represents the i^{th} peak from top

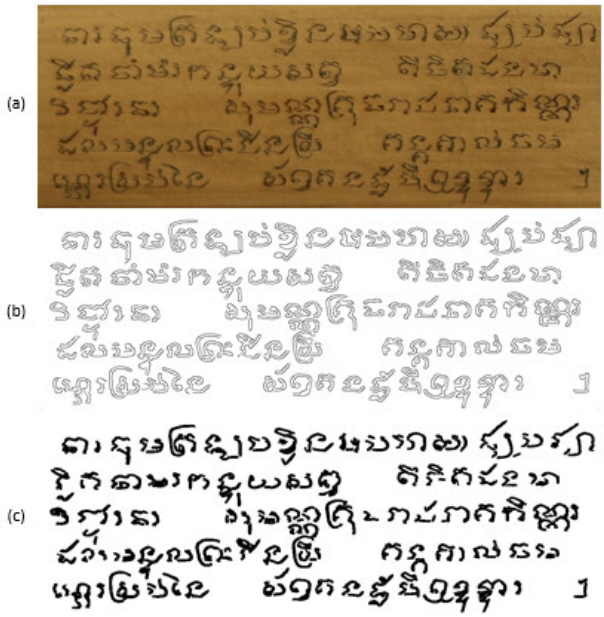

Figure 5.8: (a) Original Image, (b) Edge map using Canny edge detection, (c) Stroke map

to bottom ($0 < i < n$). We remove P_i if the distance between P_i and P_{i-1} is less than ½D_{median} where D_{median} is the median value of all distances between two adjacent peaks. The final numbers of peaks in all columns are stored in a sorted list from which we choose the value in the 95th percentile to be the number of text lines N_L in the document page.

5.2.1.3. Skew and Fluctuation Adaption (C)

To construct an estimation of medial seam of a text line, we connect corresponding medial points situated at the peaks of the Y-projection histogram of neighboring columns together. However, some peaks may be missing due to the fact that different numbers of peaks can be found in different columns. To guarantee a smooth connection between peaks, new medial points may be added. We want to have eventually a $N_L \times N_C$ grid of medial points. Let's denote $M_{l,c}$ a medial point on text line l in column c with $0 \leq l < N_L$ and $0 \leq c < N_C$ and $M_{l,c}^x, M_{l,c}^y$ its x-coordinate and y-coordinate respectively. Each medial point can be added as follows. First, relative to the starting point of the text region, each medial point should be placed horizontally at the center of its column ($M_{l,c}^x = c.W_C + ½W_C$). Then we select a starting column j whose Y-projection profile consists of exactly N_L peaks. Among multiple instances of such columns, we pick the one with the highest average peak variance and place one medial point at each peak. To the left of column j, we iterate one column at a time (column k, $k = j - 1, j - 2, \dots, 0$) to find the closest peaks to the corresponding peaks of the column to its right (column $k + 1$) and place a medial point there. If no peak is found, we place a medial point at the same y-coordinate position of the corresponding medial point in column $k + 1$ ($M_{l,k}^y = M_{l,k+1}^y$). We also maintain the distance between two adjacent medial points from the same column to be approximately equal to the common distance D_{median}. To do so, if the distance from the previously placed medial point $M_{l-1,k}$ is less than ½D_{median}, the new medial point $M_{l,k}$ is instead placed at $M_{l,k}^y = M_{l-1,k}^y + D_{median}$. We place the medial points one by one from top to bottom until we reach the total number of N_L points per column. The placement can be expressed by the pseudo code shown in

Algorithm 5.1. Similarly, to the right of column j, medial points are added from top to bottom one column at a time (column k, $k = j + 1, j + 2, \dots, N_C - 1$).

While the above procedure already gives an acceptable medial seam of text lines it is important to further adapt the seams accurately to the skew and fluctuation of text lines. For that purpose, a competitive algorithm is used in order to adapt the vertical position of medial points to the surrounding connected components. Figure 5.9 illustrates these two successive skew adaption steps.

Let us denote y_{C_i} the y-coordinate of the center of mass of a connected component $C_i, 0 \leq i < N_{CC}$ where N_{CC} is the total number of connected components extracted. The grid-like set of medial points extracted as mentioned above plays the role of centroids for the competitive learning algorithm. The so-called winning centroid with respect to each component C_i, is the medial point $M_{l,c}$ whose vertical distance to C_i is the smallest:

$$|y_{C_i} - M_{l,c}^{y}\ | \leq |y_{C_i} - M_{r,c}^{y}\ |\ \forall r \in [0, N_L[\qquad (5.9)$$

After being selected as a winner, the winning medial point is moved by a fraction of its distance towards the connected component C_i.

$$M_{l,c}^{y} \leftarrow M_{l,c}^{y} + \alpha_k . \beta_{c,C_i} . (y_{C_i} - M_{l,c}^{y}) \qquad (5.10)$$

for $k = j - 1$ down to 0
  for $l = 0$ to $N_L - 1$
    place $M_{l,k}$ at the peak closest to $M_{l,k+1}$
    if $M_{l,k}$ is NOT valid
      $M_{l,k}^{y} = M_{l,k+1}^{y}$
    end if
    if $l > 0$ **and** $M_{l,k}^{y} - M_{l-1,k}^{y} < \frac{1}{2} D_{median}$
      $M_{l,k}^{y} = M_{l-1,k}^{y} + D_{median}$
    end if
  end for
end for

Algorithm 5.1: Pseudo code illustrating the placement of medial points

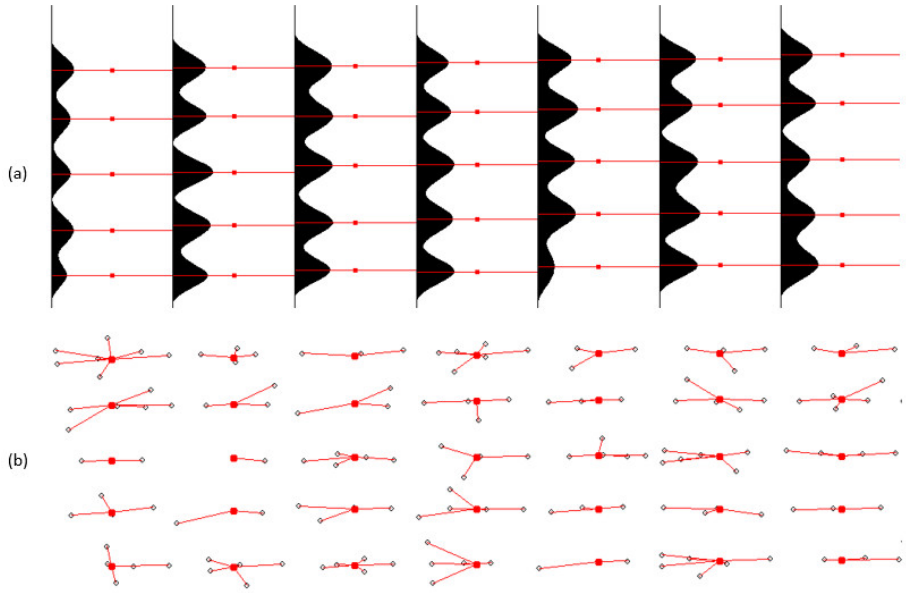

Figure 5.9: (a) Smooth projection profiles and the estimated medial points (red dots), (b) Adaption of the medial points to surrounding connected components (black dots)

After each iteration k, the learning step α_k is forced to decrease towards zero ($\alpha_k = \alpha_{k-1}/(1 + \alpha_{k-1})$) from an initial value α_0 (α_0 is set to 0.15 in our experiment) so that we can halt the algorithm. In order to take into account, the horizontal distance between the component C_i and medial point, we introduce a weight β: the value of β is large if the connected component is close to the medial point and decreases with the distance to the latter. This means that the components at a distance far away do not have much effect on the movement of the medial point. We use a Gaussian function (mean $\mu = M_{l,c}^{x}$) for the value of β_c which is normalized so that $0 < \beta_c \leq 1$.

5.2.1.4. Line Boundary Creation (D)

To define the boundary between two adjacent lines, a path finding technique is used. Our path finding approach is inspired by the A* path planning method proposed by [30]. The objective of path planning is to compute the shortest path from a starting point to its destination avoiding obstacles along the way. A* is one of the path planning algorithms that minimizes the traveling costs between states (a state refers to each pixel in the sequence of neighboring pixels representing

a particular path) from the starting state to the goal state. To solve the line segmentation problem, paths separating text lines need to be traced from the left side (starting state) to right side (goal state) of the text, and the pixels which belong to text are viewed as obstacles. However due to some handwritten text components from adjacent lines being touched or over- lapped, the goal state can be unreachable. A modified A* path-planning technique is proposed here to allow the path to pass through such components.

First, separating seams between text lines are created from the already defined medial points. We denote a new set of border points $B_{l,c}$, $0 \leq l < N_L - 1$ and $0 \leq c < N_C$. It is defined as follows:

$$B_{l,c} = \frac{M_{l,c} + M_{l+1,c}}{2} \qquad (5.11)$$

For each separating seam, we find a sequence of $N_c - 1$ paths, and each path starts from the starting state s_1 at $B_{l,k}$ to the goal state s_n at $B_{l,k+1}$, $k \in [0, N_C - 1[$. To emphasize the text foreground, the original grayscale image of the manuscript page is filtered using Sobel operator. The inverted gradient map I_{grad} of the image is used so that the pixel intensity of the foreground text is less than the one of its background. Two cost functions are computed and combined to obtain the final traveling cost $C(s_i, s_j)$ between states. If we denote $s_{1,a}, s_{2,a}, \dots, s_{n_a,a}$ as the sequence of states traversed by path p_a, the goal of the path finding algorithm is to determine the optimal path $p_{optimal}$ with the minimum total traveling cost:

$$p_{optimal} = arg \min_{p_a} \sum_{i=1}^{n_a - 1} C\left(s_{i,a}, s_{i+1,a}\right) \qquad (5.12)$$

The two function costs are described as follows:

Intensity difference cost function $D(s_i, s_j)$: This function enforces the path to avoid passing through foreground pixels. The function returns a large value when the pixel intensity abruptly changes from high to low demonstrating a situation when an edge of a foreground text stroke is encountered (going from background pixels to foreground pixels). The function however does not give penalty but encourages it

when the path leaves the foreground stroke (going from foreground pixels to background pixels). The cost function $D(s_i, s_j)$ is computed by:

$$D(s_i, s_j) = max(D_I . |D_I|, 0) \quad (5.13)$$

$$where\ D_I = I_{grad}(s_i^x, s_i^y) - I_{grad}(s_j^x, s_j^y) \quad (5.14)$$

Vertical distance cost function $V(s_j)$: This function reassures that the path does not deviate too far from the seam line preventing the path from jumping up or down the entire line region. It is the vertical distance from the state s_j to the slanted seam line constructed from the two points at s_1 and s_n. This cost function is defined as:

$$V(s_j) = \left| s_j^y - s_1^y - \frac{(s_j^x - s_1^x)(s_n^y - s_1^y)}{s_n^x - s_1^x} \right| \quad (5.15)$$

In order to connect continuously consecutive paths, we set the starting state s_1 of the next path in the sequence to be at the same position as the goal state s_n from the previous path. We combine these two cost functions to achieve the final traveling cost $C(s_i, s_j)$ defined as follows:

$$C(s_i, s_j) = c_d . D(s_i, s_j) + V(s_j) \quad (5.16)$$

where c_d is a parameter which controls how the path choose its next state to traverse. A large value of c_d means that the path would prefer staying far away from the seam line rather than passing through foreground text pixels. In our experiment, a small validation set will be used to define c_d. Since the path traverses through states from left to right, instead of computing the cost functions for neighbor pixels in all eight directions, only five directional steps are considered: South, South-East, East, North-East, and North. Figure 5.10 shows an example of an optimal path going from the start state s_1 to the goal state s_n.

5.2.2. Experiments and Results

5.2.2.1. Evaluation Metrics

In order to evaluate the performance of the proposed method and also to compare it with existing methods in the literature, we adopt the

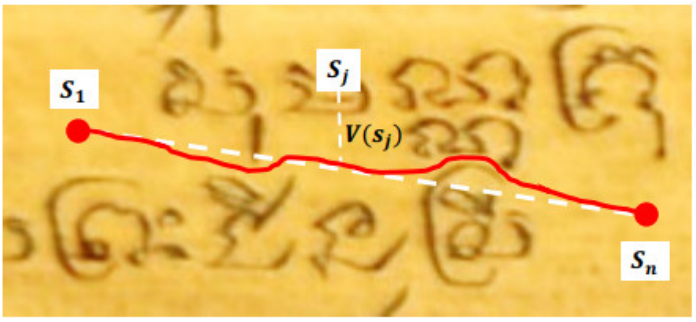

Figure 5.10: An example of an optimal path going from the start state s_1 to the goal state s_n

measures used in the ICDAR2013 Handwriting Segmentation Competition [101]. According to the evaluation method, we count the number of one-to-one matches between text lines detected by the approach and the text lines in the ground truth. The matching score is computed as follows:

$$MatchScore(i,j) = \frac{T(G_j \cap R_i \cap I_{IC})}{T\left((G_j \cup R_i) \cap I_{IC}\right)} \qquad (5.17)$$

where $T(s)$ is function that counts the number of points in set s; G_j is the set of all points inside the union of all polygon regions of isolated characters in the ground truth belonging to text line j; R_i the set of all points inside the region of result text line i; and I_{IC} the set of all points inside the union of all polygon regions of ground truth isolated characters in the whole document page.

We consider a region pair to be a one-to-one match only if the matching score is above an acceptance threshold T_a. Let's assume N to be the number of text lines found in the ground truth, M to be the number of detected text lines by the approach, and $o2o$ to be the number of one-to-one match pairs, then the detection rate (DR) and recognition accuracy (RA) are defined as:

$$DR = \frac{o2o}{N} \qquad (5.18)$$

$$RA = \frac{o2o}{M} \qquad (5.19)$$

By combining DR and RA, we obtain the evaluation metric F-measure score FM:

$$FM = \frac{2.DR.RA}{DR+RA} \quad (5.20)$$

5.2.2.2. Experiment 1

In this first experiment, we conduct an evaluation only on image samples from the SleukRith Set. A subset of 110 pages of digitized manuscript images randomly selected. The line segmentation ground truth can be constructed as follows. As described in Chapter 4.2, in order to first segment and annotate a manuscript page into small image patches representing each individual character, a polygon boundary enclosing the character needs to be drawn manually. The human ground truther is required to dot out vertices of the polygon one by one until a proper boundary is formed. After all characters in the page are manually annotated, they may be grouped together and be inserted into a line. The union of the polygon areas of all isolated characters in a line represents the total region of that line.

Among the 110 pages of the SleukRith Set, 10 pages are chosen arbitrarily and are used for parameter tuning (the optimal value of c_d=3 is found). We then apply the proposed approach to the other 100 pages (the total number of text lines in the ground truth $N = 476$). For comparison, the same subsets are also used for the methods proposed by [36] and by [37]. The evaluation results using the acceptance threshold $T_a = 0.9$ are illustrated in Table 5.3. As it can be observed from Table 5.3, the proposed method outperforms the other approaches by a large margin. Some results of line segmentation using the proposed method dealing with skew, fluctuation, and discontinuity of text lines are given in Figure 5.11.

Table 5.3 Result of the performance evaluation of line segmentation methods (Experiment 1)

Method	M	o2o	DR (%)	RA (%)	FM (%)
Method in [36]	665	356	53.53	74.79	62.40

Method in [37]	505	157	31.09	32.98	32.01
Proposed method	484	446	92.15	93.70	92.92

5.2.2.3. Experiment 2

In the second experiment, to showcase that the proposed approach is able to generalize to other datasets, we apply it on a larger dataset of text lines extracted from the SleukRith set as well as a dataset which is a collection of low-resolution images from EFEO database. We also include datasets generated from Balinese and Sundanese manuscripts. All palm leaf manuscript datasets used in this experiment are presented in Table 5.4.

Khmer 1 represents the collection of images from EFEO database. A semi-automatic scheme is used to construct a ground truth for this set. A set of medial points for each text are generated automatically on the binarization ground truth of the page image. Then those points can be moved up or down with a tool to fit to the skew and fluctuation of the real text lines. We also note touching components spreading over multiple lines and the locations where they can be separated.

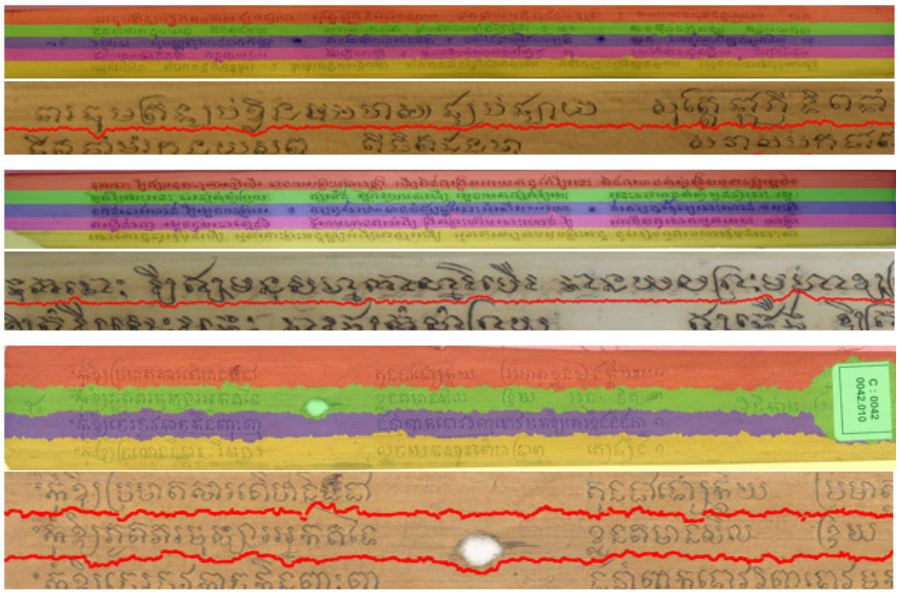

Figure 5.11: Segmentation results of the proposed approach (pairs of whole segmented manuscript page and zoomed out area with medial seams marked in red)

Khmer 2 is constructed similarly to the dataset used in Experiment 1; however, we double the number of pages in this new dataset. The text line segmentation ground truth data for Balinese and Sundanese manuscripts have been generated by hand based on the binarized ground truth images constructed on AMADI_LontarSet and Sunda dataset respectively [92, 93].

Table 5.4: Palm leaf manuscript datasets for text line segmentation task (Experiment 2)

Manuscripts	Pages	Text Lines	Dataset
Khmer 1	43 pages	191 text lines	Extracted from EFEO
Khmer 2	200 pages	971 text lines	Extracted from SleukRith Set
Balinese 1	35 pages	140 text lines	Extracted from AMADI_LontarSet
Balinese 2	Bali-2.1: 47 pages Bali-2.2: 49 pages	181 text lines 182 text lines	Extracted from AMADI_LontarSet
Sundanese 1	12 pages	46 text lines	Extracted from Sunda Dataset
Sundanese 2	61 pages	242 text lines	Extracted from Sunda Dataset

In this experiment, the proposed method and the seam carving method, which is the runner up according to the results from Experiment 1, are applied on all datasets. The experimental results are presented in Table 5.5. According to these results, both methods performs sufficiently well for most datasets except Khmer 1. This is because all images in this set are of low quality due to the fact that they are digitized from microfilms. Nevertheless, the proposed method still proves to achieve better results than the seam carving method on all datasets of palm leaf manuscripts in our experiment. The main difference between these two approaches is that instead of finding an optimal separating path within an area constrained by medial seam locations of two adjacent lines (in the seam carving method), the proposed approach tries to find a path close to an estimated straight seam line section. These line sections already

represent well enough the seam borders between two neighboring lines, so they can be considered as a better guide for finding good paths, hence producing better results.

One common error that we encounter for both methods is in the medial position computation stage. Detecting correct medial positions of text lines is crucial for the path finding stage of the methods. In our experiment, we noticed that some parameters play an important role. For instance, the number of columns/slices r of the seam carving method and the high and low thresholding values of the edge detection algorithm in the proposed approach. In order to select these parameters, similar to Experiment 1, a validation set consisting of five random pages from each dataset is used. The optimal values of the parameters are then empirically selected based on the results from this validation set.

Table 5.5: Experimental results for text line segmentation task (Experiment 2)

Method	Manuscript	N	M	o2o	DR (%)	RA (%)	FM (%)
Seam Carving Method in [36]	Khmer 1	191	145	57	29.84	39.31	33.92
	Khmer 2	971	1046	845	87.02	80.78	83.78
	Balinese 1	140	167	128	91.42	76.64	83.38
	Bali-2.1	181	210	163	90.05	77.61	83.37
	Bali-2.2	182	219	161	88.46	73.51	80.29
	Sundanese 1	46	43	36	78.26	83.72	80.89
	Sundanese 2	242	257	218	90.08	84.82	87.37
Proposed method	Khmer 1	191	169	118	61.78	69.82	65.55
	Khmer 2	971	990	910	93.71	91.91	92.80
	Balinese 1	140	143	132	94.28	92.30	93.28
	Bali-2.1	181	188	159	87.84	84.57	86.17
	Bali-2.2	182	191	164	90.10	85.86	87.93
	Sundanese 1	46	50	41	89.13	82.00	85.41
	Sundanese 2	242	253	222	91.73	87.74	89.69

6. Feasibility Study: Glyph Recognition and Localization using Deep Learning

This chapter presents a study about the feasibility of using deep learning approaches to solve handwritten text recognition problems on Khmer palm leaf documents. Trial experimentations are conducted on two basic DIA tasks: isolated glyph (or character) recognition (Chapter 6.1) and glyph localization in word images (Chapter 6.2). We also introduce in the proposed glyph localization model a special type of feature maps called "glyph-class map" (or GCM in short) which is able to store spatial information as well as the identity of all glyphs in the text image. The glyph localization problem now becomes GCM generation.

As illustrated in Figure 6.1, complete systems to recognize text on short word image patches will be developed using the outcomes from the feasibility study presented in this chapter. The detailed description as well as experimental evaluations of those systems will be explored in Chapter 7.

6.1. Isolated Glyph Recognition

The first attempt in using neural networks to solve the text recognition problem for Khmer palm leaf manuscripts is presented here. Before

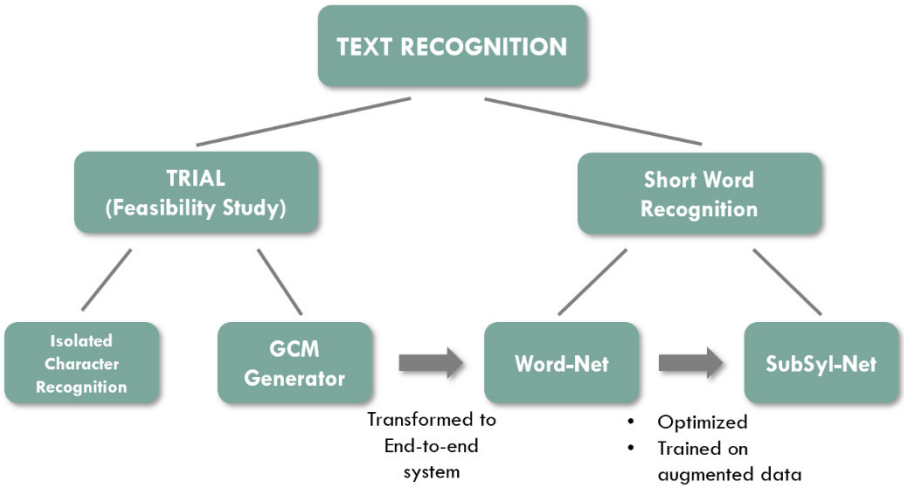

Figure 6.1: Overview of the workflow of the text recognition module

getting into a more challenging task of recognizing text images of whole text lines, individual or isolated glyph recognition is studied first. Isolated glyph recognition problem is a task that takes an input image patch containing a single glyph or symbol and predicts the class (or probabilities of all classes) that glyph in the image belongs to. For this task, we present four neural network architectures. The choice of the hyper parameters such as the number of layers as well as the number of hidden units of each model presented here is empirical. In other words, we focus on the feasibility of different types of neural network architectures (see Chapter 3.2.2) on the recognition problem of Khmer palm leaf manuscripts rather than a computationally intensive fine tuning of their structure.

6.1.1. Description of the Networks

The first network (Figure 6.2) is CNN based and is composed of two pairs of convolutional (12 and 24 of 5 by 5 filters) and max pooling layers (2 by 2 window size with 2 by 2 strides for both layers). The output from each convolutional part is activated by Relu non-linearity. A drop out with dropped probability of 0.2 is applied after each max pooling. The output from the last max pooling is flattened and followed by a fully connected layer (with 1024 hidden units) also activated by Relu. A drop out with dropped probability of 0.6 is also applied afterward. Finally, the final output with a softmax activation is produced.

The second network (Figure 6.3) is recurrent. First, the input image is transformed into a sequence of one-pixel columns. Each column is then fed into two layers of LSTM cells with 512 hidden units each. The last time step output from the last LSTM layer is connected to the final output layer which is then activated with a softmax function.

The third network (Figure 6.4) is also RNN based. However, the input image is transformed simultaneously into a sequence of one-pixel columns and a sequence of one-pixel rows. Each sequence is fed separately into two distinct pairs of LSTM layers which are similar to the ones in the second network. The last time step output from the last

layer of each pair is concatenated before being fed into the softmax activated final output layer.

The fourth network is a combination of the convolutional and max pooling part in the first network and the recurrent part in the third network (Figure 6.5). The input is first fed into the two convolutional and max pooling pairs. The output from this first two layers is split row wise and column wise along the height and width dimension respectively (the depth or channel dimension is flattened). Similar to

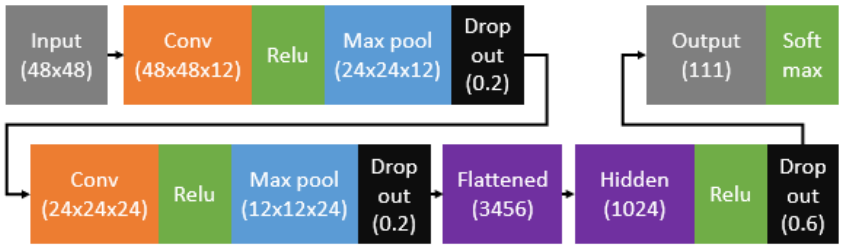

Figure 6.2: CNN based network

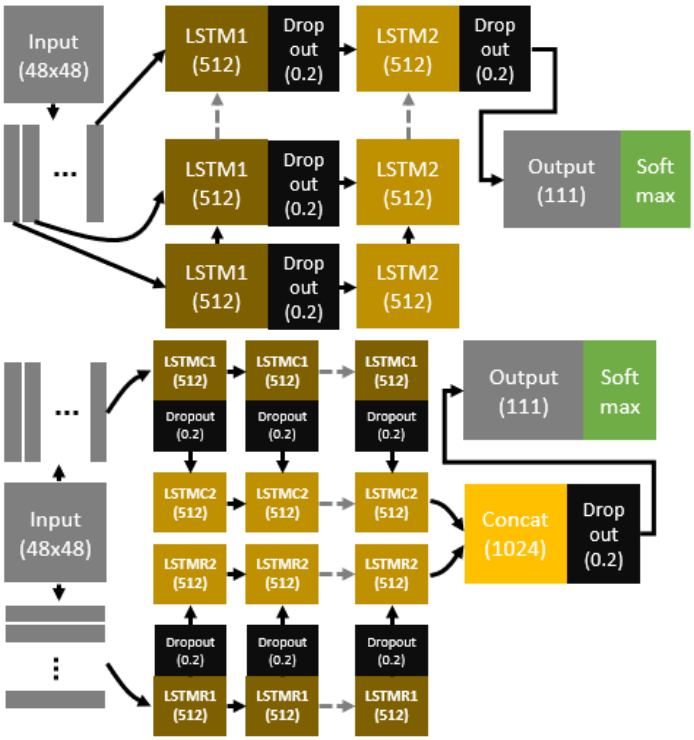

Figure 6.4: Column wise and row wise LSTM

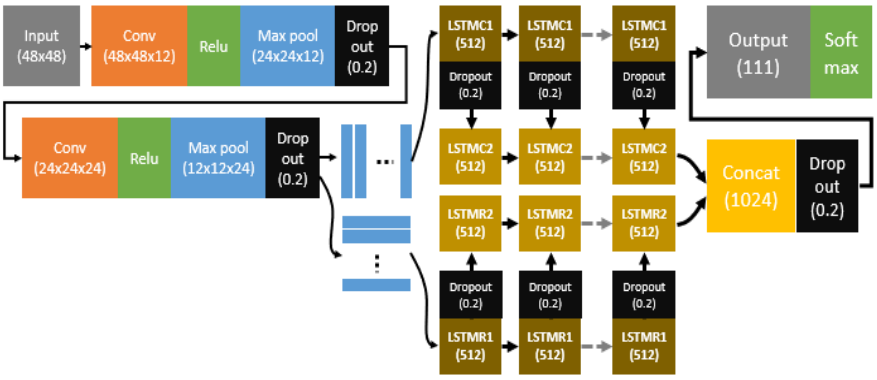

Figure 6.5: A combination of convolutional and recurrent neural network

the third network, we feed the row wise and column wise sequences to two different two-layer LSTMs (again with 512 hidden units in each layer). The outputs from the last time step of the last layer of the two LSTMs are concatenated and finally are followed by the last output layer activated by a softmax function. Like previous architectures, a drop out (with dropped probability of 0.2) is applied after each max pooling and also after each LSTM layer.

6.1.2. Experiments and Results

6.1.2.1. Datasets

The dataset for this task consists of sample images of isolated character patches extracted from different manuscript pages in SleukRith set. The glyph images are grouped by their class label. We do not take into account small symbols such as punctuations and diacritics and also remove symbols with too few occurrences. The resulting set consists of a total number of 111 classes. Each image patch is gray scaled, resized to be 48 by 48 pixels, and then normalized using histogram stretching technique (Figure 6.6).

We also extend the evaluation of the proposed methods to the datasets extracted from Indonesian manuscripts: Balinese script dataset and Sundanese script dataset. Figure 6.7 and Figure 6.8 show respectively some sample images of Balinese and Sundanese handwritten characters. Table 6.1 summarizes the datasets on all languages used for this task.

Table 6.1: Palm leaf manuscript datasets for isolated character/glyph recognition task

Manuscripts	Classes	Train	Test	Dataset
Khmer	111	113,206	90,669	SleukRith Set
Balinese	133	11,710	7,673	AMADI_LontarSet
Sundanese	60	4,555	2,816	Sunda Dataset

6.1.2.2. Experiment Procedure and Evaluation Protocols

A multi-class cross-entropy loss is used as the loss function for this task since it is suitable and effective for multi-class classification problems. For each sample i, the loss is computed by

$$L_i = \sum_{c=0}^{N_{class}-1} y_{i,c}\, log(p_{i,c}) \qquad (6.1)$$

where $y_{i,c}$ is a binary indicator if the class label c is the correct classification for the sample i (1 if it is correct or 0 otherwise), $p_{i,c}$ is the predicted probability by the network that input sample i belongs to class label c, and N_{class} is the total number of classes. The final loss is the sum of losses from all samples in the batch.

During training, the loss function of each network is minimized using Adam optimizer [102] with initial learning rate of 0.001. Weight parameters in all networks are initialized using a truncated normal distribution with a standard deviation of 0.1 while biases are initialized with constant values of 0.1. The network is trained per mini batch basis (in our experiment we choose a batch size of 300). The training set starts with all samples being shuffled, and after a competition of each epoch, the order of the samples in the training set is then again reshuffled. For

Figure 6.6: (a). Original image, (b). Gray scaled and resized, (c). Normalized

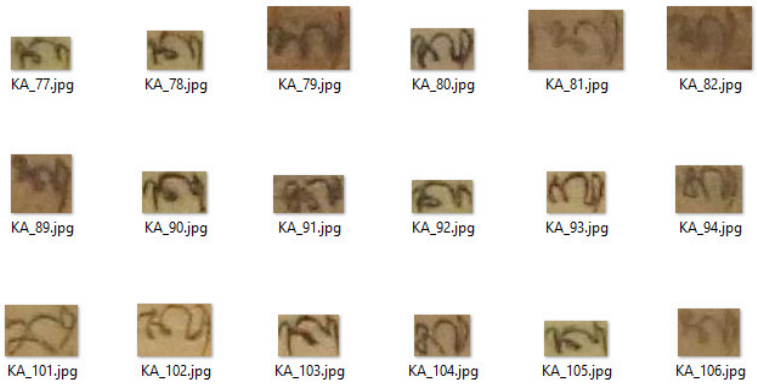

Figure 6.7: Sample images of Balinese characters

Figure 6.8: Sample images of Sundanese characters

every n_{iter} of iterations, we calculate the average loss of all batches and stop the training if the average loss does not improve for N_{test} consecutive tests. In our experiments we choose n_{iter} to be 50 and N_{test} to be 10.

Following the evaluation method from ICFHR competition [8], the recognition rate i.e. the percentage of correctly classified samples over the test samples: $N_{correct}/N_{total}$ is calculated, where $N_{correct}$ is the number of correctly recognized samples, and N_{total} is the total number of the test samples.

6.1.2.3. Results and Discussion

Since the beginning period of pattern recognition research, many feature extractions methods for character recognition have been

presented in the literature. We, therefore, also perform a comparison study with a handcrafted feature extraction approach [103] mentioned in Chapter 3.2.1. A neural network is used as a classifier which takes the feature vectors as input and predict the probability of all character classes. Additional improvement is performed by applying unsupervised learning based on K-means clustering to calculate the initial weight for the neural network training phase from the cluster centers of all feature vectors.

The experimental results of this evaluation are presented in Table 6.2. According to these results, the proposed networks (except the column-wise RNN network) perform similarly to the handcrafted feature extraction approach on Balinese and Sundanese. However, the proposed networks (again except the column-wise RNN) outperform the handcrafted one on Khmer. This is due to the size of the datasets since multi-layer neural networks require a large amount of data to train which is not the case for Balinese and Sundanese isolated character datasets. It can also be noted from the results that even being purely a recurrent network, which is more suitable in sequence modeling rather than a single object classification, the column-wise network as well as the column-row-wise network preform sufficiently well in recognizing isolated characters on palm leaf documents. The latter produces a slightly better result since it uses sequential information along both horizontal and vertical axes. It is also shown that, utilizing both the convolutional and recurrent modules is able to reach a lower error rate (compared to the CNN based only network) since on top of using convolutional modules in the shallow layers to extract principle features from the character images, the network also uses the column wise and row wise sequential information in the deeper layers. This illustrates that combining convolutional with recurrent module is a powerful technique to classify handwritten characters written on palm leaves.

Table 6.2: Experimental results for isolated character/glyph recognition task (in % recognition rate)

Methods	Khmer	Balinese	Sundanese
Handcrafted Feature (HoG-NPW-Kirsch-Zoning) with UFL + NN [103]	92.44	85.63	79.33
Purely CNN based	93.71	85.46	79.83
Column-wise RNN network	91.51	81.38	67.72
Column-wise and row-wise RNN network	93.00	84.33	71.16
Convolutional recurrent network	94.99	84.65	80.26

6.2. Glyph Localization in Word Images

In this section, we look into a more challenging problem: glyph localizing in word images. This is a sequence learning task which takes as input an image containing a handwritten text and returns as output the spatial location and the identity of all glyphs in that text. According to the experimental results from the previous section, utilizing a combination of convolutional and recurrent blocks is very efficient in recognizing Khmer handwritten characters on palm leaf manuscripts, and it is the direction that we go further in this section.

6.2.1. Glyph-class Map (GCM)

To take into account the character annotation information, for each word, a glyph-class map (GCM) is built. The word image patch is divided into grid like cells of c_h by c_w pixels where c_h and c_w are the height and the width of each cell respectively. Resizing may be applied to ensure that all cells are of equal size (after the resizing, the width of the word image patch must be divisible by c_w, and the height must be divisible by c_h). Each cell of the GCM is then assigned to one and only one glyph-class which contains the most pixels in that cell. From all the annotated words in the dataset, 134 glyph-classes are found including one additional token class for cells containing only blank space or background pixels. An example of how a glyph-class map is built is shown in Figure 6.9.

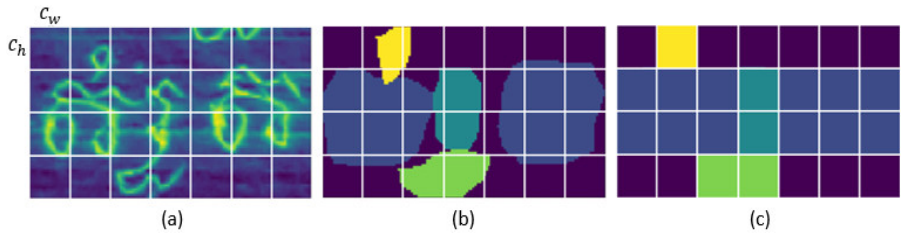

Figure 6.9: (a). Original word image patch, (b). Annotated character information in the word: polygon boundaries of all characters, (c). Glyph-class map

6.2.2. GCM Generator

The objective of the approaches for this task is just to predict the glyph-class map of the input word image patch which means to classify each cell of the input patch to its corresponding glyph-class rather than to output the text transcription of the input word image. Again, here we focus on the feasibility of different types of neural network architectures especially the choice between one-dimensional and two-dimensional RNNs.

Two network architectures are presented. Figure 6.10 shows the general architecture of both networks. The two models consist of the same two beginning convolutional layers (the filter size is 5 by 5 for both layers and the numbers of filters are 12 and 24 respectively). A max pooling layer of stride 2 follows, so the output gives rise to a $½I_h \times ½I_w \times D$ feature map where I_h and I_w are the height and the width of the input image (after possible paddings) and $D = 24$ the depth of the feature map which equals to the number of filters of the last convolutional layer. To match the number of cells in the character-class map ($n_{row} \times n_{col}$ cells where $n_{row} = I_h/c_h$ and $n_{col} = I_w/c_w$), the feature map is also divided into grid like cells where each cell is of equal size of $½c_h$ by $½c_w$ (c_h and c_w must be divisible by 2 and in our experiments, we choose c_h and c_w to be $c_h = c_w = 8$). Therefore, the final dimensions of the feature map are now $n_{row} \times n_{col} \times n_f$ where $n_f = ½c_h . ½c_w . D$.

The first network (let's call it Trial-Net1) utilizes a one-dimensional RNN (1D-RNN) architecture (Figure 6.11). Inspired by the architecture in [87], all four directions are taken into account: left to right, right to

left, top to bottom, and bottom to top. To keep the architecture as 1D, the feature map cells are flattened column wise or row wise to generate four one-dimensional sequences of length $n_{row}.n_{col}$ where each sequence corresponds to each direction mentioned above. All sequences are then fed into one shared LSTM layer with 512 hidden units to produce four output sequences also of length $n_{row}.n_{col}$ each of which is then reshaped back to be a grid of $n_{row} \times n_{col}$. The four grids are concatenated together along the feature axis to form a single grid of features ($n_{row} \times n_{col} \times 2048$). The next layers of the network are a fully connected layer with 1024 hidden units followed by a softmax activated final output layer to produce the predicted character-class map ($n_{row} \times n_{col} \times n_{class}$ where $n_{class} = 134$).

Due to the characteristics of Khmer writing especially how characters are positioned to form a word, the second network (Trial-Net2) takes into consideration both spatial dimensions of the image. For this reason, the proposed network uses a two-dimensional RNN (Figure 6.12). Unlike the first network, diagonal directions are considered instead. Four grids of cells are produced from the input feature map to represent each of the four diagonal directions: top-left to bottom-right, bottom-right to top-left, top-right to bottom-left, and bottom-left to top-right. They share a 2D LSTM layer (whose previous states are given by the

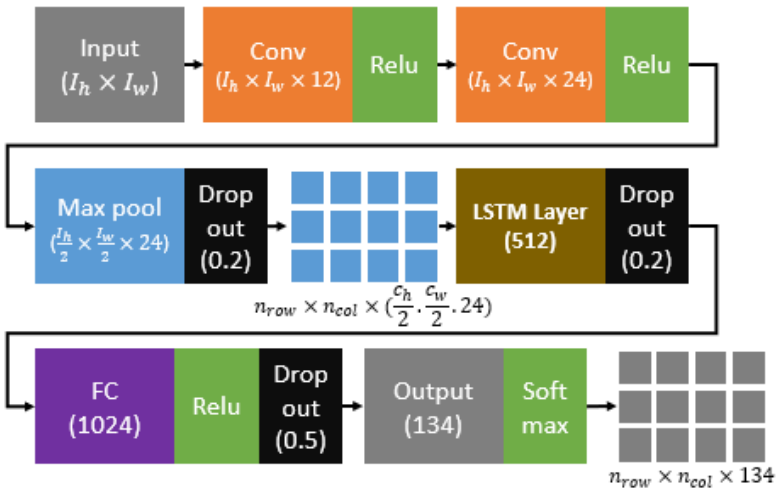

Figure 6.10: General architecture of the networks in the first trial

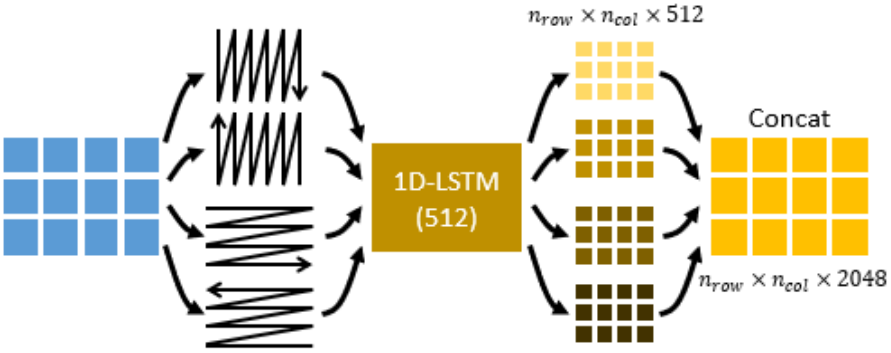

Figure 6.11: Architecture of the 1D-LSTM layer (Trial-Net1)

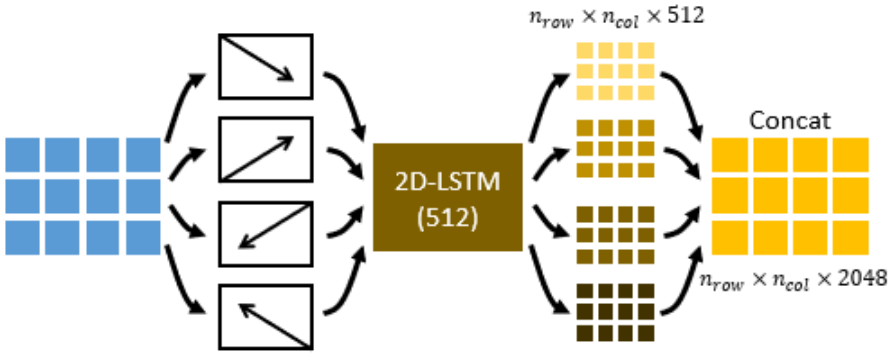

Figure 6.12: Architecture of the 2D-LSTM layer (Trial-Net2)

previous cells along both horizontal and vertical axes) also with 512 hidden units to output four grids of the same size ($n_{row} \times n_{col} \times 512$) which are then concatenated along the depth or feature axis to get a final grid before feeding it to a fully connected layer and then to the final output layer similar to the architecture of the first network.

We regularize both networks by applying dropouts with dropped probability of 0.2 after the max pooling and of 0.2 and 0.5 before and after the fully connected layer respectively. Since the text image can be big in width which would produce a very long sequence for the LSTM layer, we also use gradient clipping to prevent exploding gradients.

For efficient training, input word image patches are sorted by their width and then are batched together so that all image samples in the same batch have similar width. This saves memory space and

computing time by eliminating unnecessary large paddings (we may still need to apply small paddings so that all input images have their widths equal to the maximum width in the batch). After each epoch, the reshuffling is done on the order of batches instead of the order of every single samples. Figure 6.13 illustrates this batching mechanism.

6.2.3. Experiments and Results

6.2.3.1. Datasets

The dataset used to train the Trial-Net1 and Trial-Net2 is generated from SleukRith set. It consists of 24,009 samples of word image patches, their corresponding ground truth GCM, and their word transcriptions. The dataset is divided into three parts: around 65% for training, 5% for validating, and 30% for testing. All word image patches are in grayscale (only one-color channel) and are normalized by scaling so that they are of the same height (72 pixels) but still with variable width.

6.2.3.2. Evaluation Protocols

We use top k error rate measurement to evaluate the performance of the Trial-Net1 and Trial-Net2. Each cell of the target GCM is predicted by the networks. The error rate of one sample word image is the number of incorrectly identified cells over the total number of cells in that image. We obtain the final error rate of all samples in the test set by

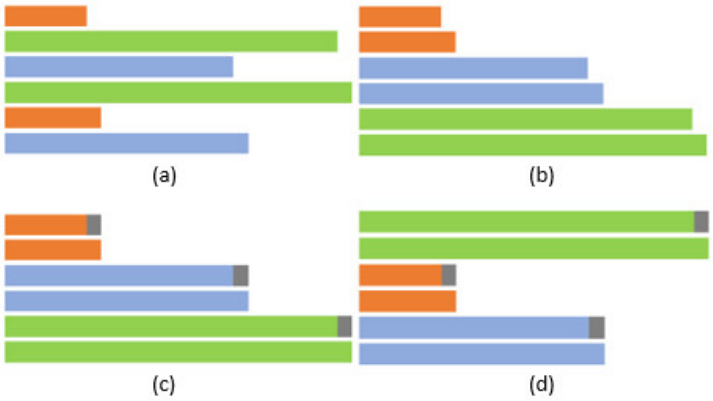

Figure 6.13: (a). Initial sample order, (b). Sort by the width of each sample, (c). Pad each sample to the maximum width in the batch, (d). Shuffle batch order

averaging the error rate of each sample. In this evaluation, we choose $k = 5$ and $k = 1$.

6.2.3.3. Results and Discussion

Table 7.2 shows evaluation results of the Trial-Net1 and Trial-Net2. From the results shown, we can see that the Trial-Net2 which uses two-dimensional LSTM outperforms the Trial-Net1 which uses only one-dimensional LSTM. This is to be expected since the LSTM module of the latter network acquires more information from the previous states in both vertical and horizontal axes. The error rates from both networks drop significantly between the top 1 and top 5 measurements. This illustrates the problem caused by the similarity and the ambiguity of Khmer characters. An example of a comparison between predicted GCM from both networks and the ground truth GCM is also given in Figure 6.14. By observing this example, we can notice that the GCM is not very sensitive to error. Connected components in the map (even with some noises) can still be used to obtain the identity of each character and its location in the text image.

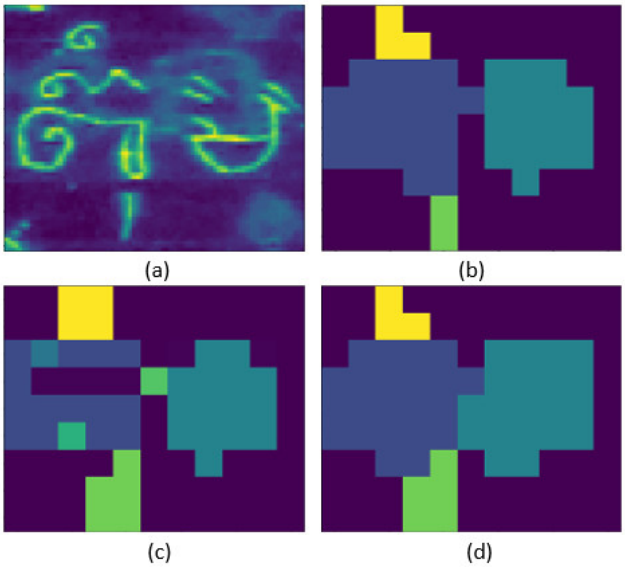

Figure 6.14: (a). Original word image, (b). Ground truth GCM, (c). Result predicted by the Trial-Net1, (d). Result predicted by the Trial-Net2

Table 6.3: Evaluation results of the Trial-Net1 and Trial-Net2

	Error Rate of the GCM Generator (%)	
	Top 1	**Top 5**
Trial-Net1	32.01	8.46
Trial-Net2	20.49	2.40

7. Text Recognition

After the trial experimentations presented in the previous chapter, we now propose an end-to-end model to recognize a handwritten text on short image patches extracted from Khmer palm leaf manuscripts. As illustrated in Figure 6.1, two novel text recognition systems following convolutional recurrent neural network architectures are proposed. The first initial system (denoted Word-Net) is used to recognize word image patches obtained from the annotated word dataset of SleukRith set (Chapter 7.1). Since the number of samples in this word dataset is still limited, a new extended dataset whose size is significantly larger is constructed. A new optimized text recognition system (denoted SubSyl-Net) is proposed in Chapter 7.2 to accommodate this new augmented dataset.

7.1. Recognition of Word Image Patches (Word-Net)

The model consists of two main modules: the GCM generator and the GCM encoder-decoder. Figure 7.1 illustrates the complete architecture of the proposed model. Both modules utilize the combination of convolutional and multi-dimensional recurrent blocks.

GCM Generator: a GCM generator takes a grayscale word image patch I with dimension $H_I \times W_I$ as input and returns a corresponding GCM of the patch as output. First, convolutional blocks are used to extract automatically the features of the word image patch. Each convolutional block is composed of a convolutional layer with a receptive field 5×5 at a fixed stride 1×1. We increase the number of feature maps from 64 to 128 and then to 256 to gradually obtain from low to higher levels of representation. To further extend the depth of the network, we also downscale the image by a factor of 2 at the end of each convolutional block by using maxpooling with kernel size 2×2 at a stride 2×2. Convolutional blocks are activated by ReLu. To regularize the model and to prevent overfitting, dropout of dropped probability $p = 0.3$ is introduced after each block. To ensure that the dimension of the output predicted by the CGM generator is identical to the ground truth CGM (i.e. $N_{row} \times N_{col}$), the feature map output from

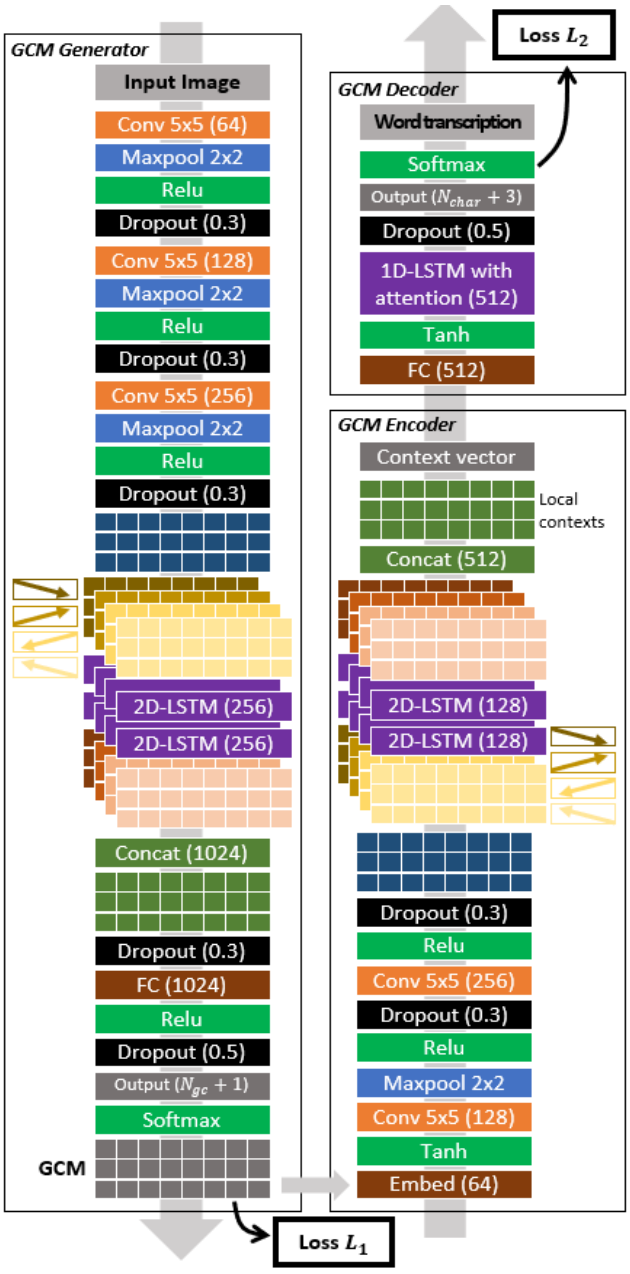

Figure 7.1: Overview of the architecture of proposed word recognition model

the convolutional blocks needs to be divided into a grid of cells of size $c'_h \times c'_w$ which can be computed as follows:

$$c'_h = \frac{c_h}{2^{N_{conv}}} \quad (7.1)$$

$$c'_w = \frac{c_w}{2^{N_{conv}}} \quad (7.2)$$

where N_{conv} is the number of convolutional blocks which is equal to 3 in the proposed architecture. We should also ensure that c_h and c_w are large enough to allow the division by $2^{N_{conv}}$. Therefore, we use $c_h = c_w = 8$ in our experiments. Each cell in the grid is then transformed into a vector by flattening out its dimension.

To take advantage of the importance of local spatial context in a two-dimensional space according to the characteristics of Khmer writing, we use multi-directional multi-dimensional LSTM (MDDLSTM) [74] in our recurrent blocks. Instead of a single hidden state from the previous time step like in the conventional one-dimensional LSTM, MDDLSTM makes use of two states each from both the vertical and horizontal axes (see Chapter 3.2.2 for more details about MDDLSTM).

To take into account all directions in the 2D space, four grids of cells are produced from the feature map grid. Those four grids represent four diagonal directions: top-left to bottom-right, bottom-left to top-right, top-right to bottom-left, and bottom-right to top-left. The four directional grids share the same two-layer block of MDDLSTM (each layer with 256 hidden units) to produce four output grids whose feature vectors in each cell are then concatenated together to transform back into a single grid of feature map. At each cell of the grid, we apply a dropout ($p = 0.3$) followed by a fully connected layer (with 1024 hidden units) activated by ReLu and another dropout ($p = 0.5$). To predict the GCM corresponding to the input word image patch, the last layer with $N_{gc} + 1$ hidden units and a softmax activation is used to output the probabilities of all glyph classes (including the class representing the background) for each cell in the predicted GCM.

GCM Encoder-Decoder: an encoder-decoder model is used to convert the GCM into final transcription of the input word image patch. This encoder-decoder module is separated into two sub-modules: an encoder and a decoder. The encoder encodes the GCM generated by the GCM

generator into a representation vector called context vector. The decoder then uses the context vector as an initial state to predict the Unicode transcription one letter at each time step.

We propose a combination of convolutional blocks and recurrent blocks as our GCM encoder. It takes as input the GCM and first reduces the dimension of the vector in each cell by passing it through an embedding layer (64 neurons) and then squash it using Tanh activation. Since the GCM contains information about the identity and the number of glyphs appearing in the word image patch and also their estimated boundary regions, two convolutional blocks are used to capture the bottom features of the map. Two main benefits of these convolutional blocks are that (1) the features extracted are useful in detecting and grouping together automatically the neighbouring cells belonging to the same glyph region without the need for handcrafted method such as connected component extraction and also that (2) the maxpooling layer down-samples the GCM dimensionality which limits the length of the input sequence to the recurrent block of the encoder to be not too long. For the purpose of regularization, dropouts are used after each convolutional block. Due to the GCM being two-dimensional, we again use MDDLSTM in the recurrent block of the GCM encoder. Similar to the description of applying MDDLSTM in the GCM generator mentioned previously, the recurrent block output four grids along four different diagonal directions. The four grids are afterwards merged back together to form the final grid with dimension $½N_{row} \times ½N_{col}$ (the GCM is down-sampled by a factor of 2 due to the maxpooling layer in the first convolutional block) which is used to compute the output context vector by averaging all its cells. The final grid of the encoder can also be referred to as the local contexts of the GCM. Both the context vector and the local contexts are sent to the decoder to be decoded into word transcription.

A GCM decoder is used to predict the next letter or character in the word transcription given the context vector generated from the encoder and all previously predicted characters. The characters to be predicted are represented by integer values C, $0 \leq C < N_{char} + 3$ where N_{char} is

the total number of Khmer Unicode characters (which are limited between U1780 and U17E9). We add three more character codes to represent the start token C_{start}, the end token C_{end}, and the unknown character C_{unk}. Before being fed to the module as input, the character representation value C is transformed into a vector by using the one hot encoding technique. For this module, we use a conventional one-dimensional LSTM as the recurrent block. Figure 7.2 shows the detailed architecture of the GCM decoder. Before becoming the initial hidden state of the LSTM, the context vector is first passed through a fully connected layer with equal number of hidden units (512) and is activated by Tanh function.

Since the generated GCM may contain multiple groups of cells representing multiple regions of glyph boundaries, each predicted character from the decoder should be conditioned on a different region of cells. Instead of relying only on a single encoded context vector, the decoder should pay its attention to particular regions in the CGM to predict efficiently the correct character at each time step. The local contexts provided also by the GCM encoder are useful in this situation. We adopt the attention mechanism proposed by [104].

Denote $s^{(ij)}$ a local context at position (i, j), $0 \leq i < ½N_{row}$ and $0 \leq j < ½N_{col}$, the attention vector at each time step t (a_t) is computed as a weighted sum of the local contexts.

$$a_t = \sum_{i=0}^{\frac{N_{row}}{2}-1} \sum_{j=0}^{\frac{N_{col}}{2}-1} \alpha_t^{(ij)} \odot s^{(ij)} \qquad (7.3)$$

The weight vector $\alpha_t^{(ij)}$ of each local context $s^{(ij)}$ is computed by

$$\alpha_t^{(ij)} = \frac{exp(e_t^{(ij)})}{\sum_{m=0}^{\frac{N_{row}}{2}-1} \sum_{n=0}^{\frac{N_{col}}{2}-1} exp(e_t^{(mn)})} \qquad (7.4)$$

$$e_t^{(ij)} = f_{att}(h_{t-1}, s^{(ij)}) \qquad (7.5)$$

where f_{att} is a small neural network with one hidden layer of 512 units

$$f_{att}\left(h_{t-1}, s^{(ij)}\right) = W_{att}\left[h_{t-1};\ s^{(ij)}\right] + b_{att} \qquad (7.6)$$

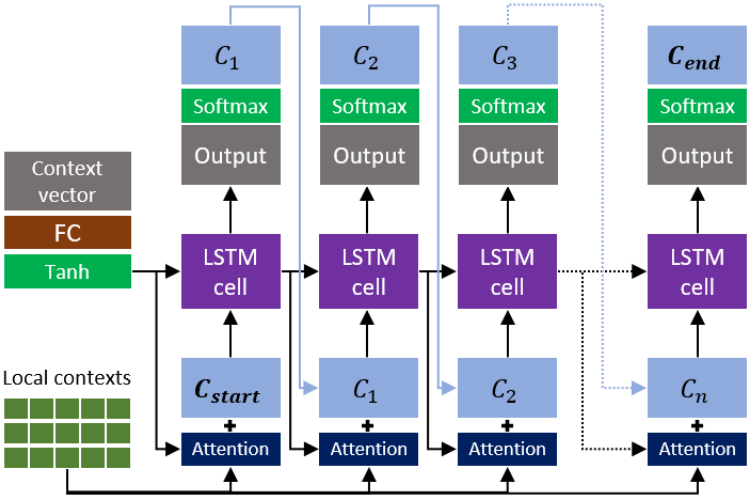

Figure 7.2: Detailed architecture of the GCM decoder

which is used to learn the weight vector $\alpha_t^{(ij)}$ at time step t in function of the previous hidden state of the decoder h_{t-1} and each local context $s^{(ij)}$. The input x_t to the LSTM is the concatenation of the one hot encoding of the character and the attention vector a_t. The decoder always has the start token C_{start} as its first input at time step $t = 0$. The current hidden state from the recurrent block is then fed into the final output layer (after applying a dropout with dropped probability $p = 0.5$), and a softmax function is applied afterwards. This sotfmax activated output is used to create the input for the next time step. The decoder stops generating new characters when the end token C_{end} is encountered or when the output transcription reaches a maximum length.

Beam Search: Instead of using greedy search i.e. choosing the character with the highest probability at each time step, we adopt the beam search with length normalization as proposed by [105]. The beam search technique maximizes the joined probability of all characters in the predicted word transcription by keeping the top k predictions as hypotheses. To not let the search prefer short transcriptions to long ones, the joined probability of each hypothesis is normalized by being divided by L_{norm} which is computed as follows:

$$L_{norm} = \frac{(\beta+L)^{\gamma}}{(\beta+1)^{\gamma}} \quad (7.7)$$

where L is the length of the predicted transcription in each hypothesis. In our experiments, we select the beam size k to be 5, and the hyper parameters β and γ are chosen to be 5 and 0.7 respectively as recommended by [105]. The hypothesis whose joined probability is the maximum is chosen as the final output transcription.

Loss Functions: To train the Word-Net, two losses are minimized. The first loss L_1 corresponds to how well the generator generates the CGM while the second loss L_2 captures the overall performance of the model to predict the final word transcription. For each sample image, those two losses are computed as follows:

$$L_1 = -\sum_{i=0}^{N_{row}-1}\sum_{j=0}^{N_{col}-1}\sum_{k=0}^{N_{gc}} y_{gc,k}^{(ij)}\, log(p_{gc,k}^{(ij)}) \quad (7.8)$$

where $p_{gc,k}^{(ij)}$ is the probability that the generator predicts that the cell at the ith row and the j^{th} column of the predicted GCM belongs to glyph class k, and $y_{gc,k}^{(ij)}$ is equal to 1 if the cell at position (i,j) of the ground truth GCM belongs to glyph class k or otherwise it is equal to 0. The second loss L_2 is computed by

$$L_2 = -\sum_{i=0}^{L-1}\sum_{k=0}^{N_{char}+2} y_{char,k}^{(i)} log\ (p_{char,k}^{(i)}) \quad (7.9)$$

where $p_{char,k}^{(i)}$ is the predicted probability of character of class k at time step i, $Y_{char}^{(i)} = [y_{char,k}^{(i)}]$ $(0 \leq k < N_{char} + 3)$ is the one hot encoding of the ith character in the ground truth transcription, and L is the length of the ground truth transcription. The total loss of the complete model is then computed by

$$L_{total} = \lambda_w L_1 + (1 - \lambda_w)L_2 \quad (7.10)$$

where λ_w ($\lambda_w \in [0,1]$) is a hyper parameter to control how generating the GCM affects the total loss of the Word-Net (see Chapter 7.4 for the choice of λ_w).

Training: During training the total loss of the function is minimized using Adam optimizer [102]. The GCM generator and the GCM encoder-decoder are pre-trained separately to minimize their corresponding losses (L_1 and L_2 respectively) using a normal distribution with standard deviation of 0.1 as initial weights and constants values of 0.1 as initial biases for all layers of the network. For the GCM encoder, the ground truth GCM is used instead as input. We also adapt the teacher forcing technique for the GCM decoder. The technique feds the characters in the ground truth word transcription to the decoder for the prediction of later outputs instead of using the predicted output from the previous time step of the decoder itself. This teacher forcing behavior forces the decoder to stay close to the ground truth sequence resulting in faster training. Periodically every five epochs, we alternatively train with teacher forcing for the first three epochs, and without it for the last two epochs. After each module converges, the complete network is then fine-tuned by minimizing the total loss as computed in Equation 7.10.

The network and its modules are trained per mini batch basis (25 samples per batch). For efficient training, input word image patches are sorted by their width and are then batched together so that all image samples in the same batch have similar width. At the start of each epoch, the order of the batch is shuffled. To ensure that all images in the same batch have the same dimension, they are rescaled to new height $\widehat{H_I}$ and width $\widehat{W_I}$:

$$\widehat{H_I} = (1 + \epsilon_H)H_I \qquad (7.11)$$

$$\widehat{W_I} = (1 + \epsilon_W)W_{I,min} \qquad (7.12)$$

where $W_{I,min}$ is the minimum width of the batch, and ϵ_H and ϵ_W are small values selected arbitrarily between $[-0.15,0.15]$. This rescaling also provides data augmentation to the training set due to the random nature of ϵ_H and ϵ_W.

For every N_{iter} of iterations, we evaluate the network on the validation set and stop the training if the evaluation result does not improve for

N_{epoch} consecutive epochs. In our experiments we select $N_{iter} = 50$ and $N_{epoch} = 5$.

7.2. Recognition of Sub-syllable Image Patches (SubSyl-Net)

By using the data augmentation technique (see Chapter 4.2.4), a new and significantly larger dataset consisting of groups of sub-syllables is constructed. To adapt to this new dataset, instead of using the Word-Net described previously, in this section we propose a completely new pipeline of text recognition system which is used to recognize a short image patch consisting of a group of glyphs and to return its corresponding text transcription (denoted SubSyl-Net). One of the reasons behind developing a new architecture for the sub-syllable dataset is due to the capacity of our current computing resource. Since the new sub-syllable dataset is significantly larger than the word dataset (see Table 7.1 for the exact number of samples in both datasets), training the previous network on this new dataset would be too time-consuming. We, therefore, keep in mind possible optimization options for this new architecture. Some optimization choices include the use of inception blocks (described in Chapter 7.2.2) instead of large convolutional layers and the substitution of multi-directional multi-dimensional LSTMs with BLSTM in the recurrent blocks (owing to their comparable performance as discussed in [106]).

As illustrated in Figure 7.3, the proposed system comprises two main modules: a generator which extracts feature maps containing the context describing the input image and a decoder which maps the variable length feature maps and interprets them into variable length output text sequences. To capture the spatial information as well as the identity of each glyph in the image patch, the generator outputs the modified version of the GCM (which will be mentioned in detail next). In addition, a second feature map is also produced by the generator to acquire extra abstract information which is used to help decode the text transcription.

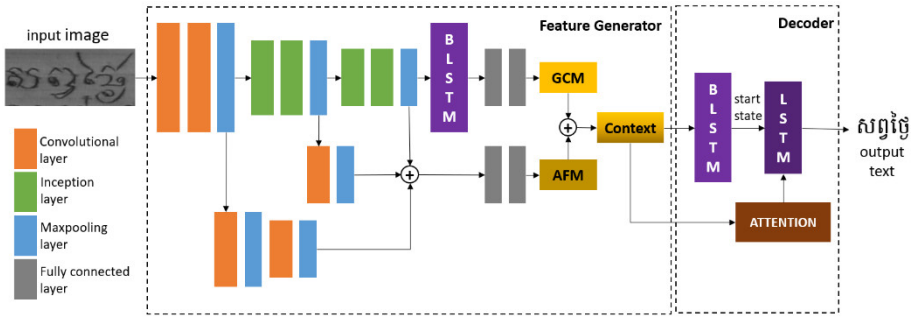

Figure 7.3: Overview of the architecture of the SubSyl-Net

7.2.1. Modified GCM

To improve the GCM mentioned previously (eliminating the hyper parameters c_h and c_w as well as their compatibility with the input image size), a modified version of the GCM is proposed. Let's suppose an image patch of a sub-syllable group I composing of n glyphs, and B_i $(0 \leq i < n)$ represents the region bound by the polygon boundary of the i^{th} glyph g_i. In each region B_i, we replace the value of each pixel by a new value v_i $(0 < v_i \leq N_{gc}$ where N_{gc} is the number of glyph classes) corresponding to the class of the glyph g_i. A new image I' with the same dimension as I is created by forming the union of all regions B_i. An additional value $(v_{blank} = 0)$ is used to fill in the background region of I' where no glyph pixels are assigned to. To obtain the final GCM, the image I' is downsampled by applying nearest-neighbor interpolation. The process of how the new GCM is constructed is illustrated in Figure 7.4.

7.2.2. Feature Generator

This component of the network takes as input a grayscale image patch of a sub-syllable group and produces two outputs: a corresponding GCM and an additional feature map. The generator starts with a convolutional block given its effective capability in extracting visual features from images. The block comprises of two basic convolutional layers to output a low-level feature map whose dimension is reduced by passing through a maxpooling layer.

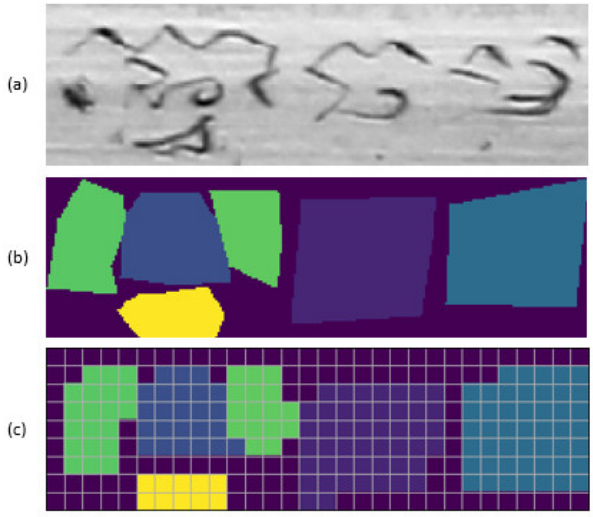

Figure 7.4: Modified version of GCM, (a) Raw image patch I, (b) Map I' containing polygon boundaries of all glyphs, (c) Downsampling I' by applying nearest neighbor interpolation to obtain the new GCM

Next we utilize two inception blocks following the previous convolutional block. Each inception block is composed of two inception layers and again at the end of the block, a maxpooling layer is applied to down sample the dimension of the output map. The proposed inception layer in this work is derived from the inception module introduced by [107]. The principle idea of the inception layer is that multiple convolutional layers with different kernel sizes in a single module can be used as a feature extractor to capture multi-scale contextual information from the input. In another word, a single inception layer contains multiple inception paths, and from these paths, feature maps of different scales are combined to approximate more complicated feature maps which can otherwise be achieved only through larger filters and more layers. The architecture of the inception layer used in our proposed system is shown in Figure 7.5.

Formally, the input image I of dimension $H_I \times W_I \times 1$ becomes a feature map $\mathcal{M}$ with dimension $H_\mathcal{M} \times W_\mathcal{M} \times F_\mathcal{M}$ after it passes through the convolutional block and then the two inception blocks (H and W represents the height and the width dimensions respectively). The input

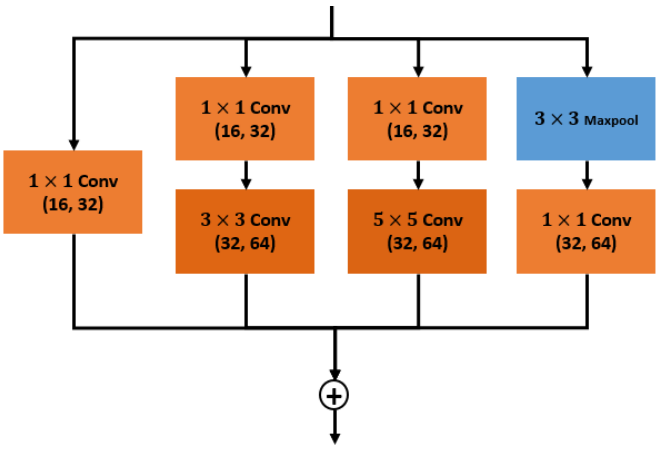

Figure 7.5: Detailed architecture of the inception layer. Values in the parenthesis are the numbers of filters (corresponding respectively to the first and the second inception blocks in the SubSyl-Net

image is grayscale (only one channel) hence its third dimension is 1, and F_M is the size of the output feature. The height and width dimension of the output feature map is reduced in size by a factor of $2^{N_{pool}}$

$$H_M = H_I/2^{N_{pool}} \tag{7.13}$$

$$W_M = W_I/2^{N_{pool}} \tag{7.14}$$

due to the application of maxpooling, where N_{pool} is the number of the 2×2 maxpooling layers used in the generator.

In order to also encode the temporal context which is significant especially in the complex writing such as Khmer whose letters are position dependent, the feature map $\mathcal{M}$ is transformed into a one dimensional sequence of length $H_{\mathcal{M}}.W_{\mathcal{M}}$ whose each element is a vector of size $F_{\mathcal{M}}$. To process the sequence, we utilize a Bi-directional LSTM (BLSTM), a combination of two LSTM layers which read the sequence in opposite directions to encode both forward and backward dependencies. The BLSTM outputs two sequences whose corresponding elements are concatenated to return back to a single sequence of feature vectors. Each feature vector is passed on to a fully connected layer and then to a final output layer activated by a softmax to produce the probability distribution for each glyph class. The output

sequence is finally transformed back to its previous dimension $H_{\mathcal{M}} \times W_{\mathcal{M}} \times N_{gc}$ to represent the predicted GCM where N_{gc} is a total number of glyph classes.

In addition to the GCM, we also consider an additional feature map (let's denote it AFM) which is useful in decoding the text transcription since this feature can automatically capture other information not already contained in the GCM. As illustrated in Figure 7.3, instead of adapting a new path to output such feature, we reuse the features obtained from different levels of the GCM production architecture. This new feature combines information from different levels of abstraction and depth of the generator network by concatenating the output feature map after the convolutional block and the feature maps after each of the inception block (including the map $\mathcal{M}$). Prior to the concatenation, to ensure that all feature maps are of the same dimension (height and width), we repetitively apply convolutional and maxpooling layer pair to each of the feature map (except the last map $\mathcal{M}$) until each feature map obtains the dimension of the last map $\mathcal{M}$. The final concatenation of these feature maps produces the AFM. The GCM and the feature map AFM are finally concatenated along the feature dimension and are used as the context to be decoded by the decoder to predict the corresponding text transcription of the input image.

7.2.3. Decoder and Attention Mechanism

Let's denote the context extracted from the input image by the generator as $\mathcal{H}$ (the concatenation of GCM and AFM) whose dimension is $H_{\mathcal{H}} \times W_{\mathcal{H}} \times F_{\mathcal{H}}$ where $H_{\mathcal{H}} = H_{\mathcal{M}}$, $W_{\mathcal{H}} = W_{\mathcal{M}}$, and $F_{\mathcal{H}} = F_{GCM} + F_{AFM}$ ($F_{GCM} = N_{gc}$ and F_{AFM} the size of the feature dimension of the feature map AFM). The context $\mathcal{H}$ can also be viewed as a matrix (size $H_{\mathcal{H}} \times W_{\mathcal{H}}$) of feature vectors $h_{i,j}$, $0 \leq i < H_{\mathcal{H}}$ and $0 \leq j < W_{\mathcal{H}}$. Our proposed attention-based decoder utilizes a one directional LSTM to predict the text transcription one character at a time. The decoder produces at each time step t the probability distribution over all characters $Y^{(t)}$ conditioned on the context $\mathcal{H}$ and all previously seen character distributions $Y^{(0)}, Y^{(1)}, \ldots, Y^{(t-1)}$. Again, $Y^{(t)}$ is a character probability distribution predicted by the decoder at time step t i.e.

$Y^{(t)} = [y_c^{(t)}]$ where $y_c = p(c) \in [0,1]$ is a probability that the predicted character is likely to be c, $0 \leq c < N_{char} + 3$. Here, we represent each Khmer character by an integer c, and N_{char} is the total number of characters that are used. Similar to the decoder of the Word-Net, we also include three additional special characters to represent a start token (c_{start}), an end token (c_{end}), and an unknown character (c_{unk}).

Recall that the prediction of the output character at each time step t is also conditioned on the context matrix $\mathcal{H}$. Since the context $\mathcal{H}$ maintains the spatial information of all glyphs from the input image, to efficiently predict the correct character, the decoder should concentrate specifically on a particular group of context vectors in $\mathcal{H}$ (the group of context vectors which corresponds to the region in the input image where the character is located) rather than to pay its attention equally to all context vectors in $\mathcal{H}$. At each time step t, this weighted context or so-called context with attention (denoted $\mathcal{A}^{(t)}$) is used instead of $\mathcal{H}$. Each context vector $a_{i,j}^{(t)}$ in $\mathcal{A}^{(t)}$ is the corresponding context vector $h_{i,j}$ in $\mathcal{H}$ weighted by a coefficient $\alpha_{i,j}^{(t)}$:

$$a_{i,j}^{(t)} = \alpha_{i,j}^{(t)} * h_{i,j} \qquad (7.15)$$

To compute the coefficient $\alpha_{i,j}^{(t)}$, we use a small neural network with one hidden layer whose input depends on the previous hidden state of the decoder (the hidden state at time step $t-1$) and the corresponding context vector $h_{i,j}$ in $\mathcal{H}$. The output of the network is squashed by a sigmoid function so that each $\alpha_{i,j}^{(t)} \in [0,1]$.

At the beginning of the decoding process (at time step $t = 1$), the LSTM of the decoder requires two initial states: the start cell state and the start hidden state. For the start cell state, a vector of all zeros is used. However, for the initial hidden state, to utilize the context vectors $h_{i,j}$ at every location (i, j) as well as to capture the spatial information to predict the first character of the text transcription, another BLSTM is applied. The context matrix $\mathcal{H}$ is transformed into a one-dimensional sequence of length $H_{\mathcal{H}} * W_{\mathcal{H}}$ which is used as the input sequence to the

BLSTM to produce two sequences corresponding to both forward and backward directions. The two sequences are concatenated, and finally the initial hidden state of the decoder is computed by calculating the mean of all elements in the concatenated sequence. We feed the decoder as an initial input, a one hot encoding of the start token c_{star} ($y_{c_{start}}^{(0)} = 1, y_{c \neq c_{start}}^{(0)} = 0$) concatenated with the sum of all context vectors $a^{(1)} = \sum_i \sum_j a_{i,j}^{(1)}$ in the weighted context $\mathcal{A}^{(1)}$. For the next time steps, we recursively perform the same process until the decoder produces a distribution which represents the end token c_{end}.

7.2.4. Implementation Details

In our experiments, all input image patches are scaled so that they are of the same height of 72 pixels with the aspect ratio preserved.

The generator: the convolutional block consists of two conventional layers both using 64 3×3 filters and activated by ReLU followed by a 2×2 maxpooling layer. The next module is composed of two inception blocks. The first block contains two inception layers with the same setting (number of parameters). There are also two inception layers in the second block with larger parameters (see Figure 7.5 for the numbers of filters used).

Again, after each inception block, a 2×2 maxpooling layer is applied to reduce the height and width dimensions of the output feature map by a factor of 2. All convolutional components in each inception layers use ReLu activation. We also use a BLSTM with 128 hidden units which produces an output whose feature dimension is 256 since the BLSTM returns two sequences (for both forward and backward directions). Following the recurrent module, a ReLu activated fully connected layer with 192 hidden units is used. The final output layer is activated by a softmax to generate a probability distribution over N_{gc} glyph classes. In the sub-syllable dataset constructed from SleukRith set, $N_{gc} = 149$ glyph classes (including the class representing the background) are found. To output the AFM, three feature maps from different depths are concatenated together. The first feature map is from the convolutional block which is down sampled by applying two consecutive

convolutional and maxpooling pairs. The convolutional layer in each pair consists of 64 3×3 filters while the maxpooling layer uses a 2×2 filter. The height and width of this feature map is reduced by a factor of 4 since we apply the maxpooling twice. We use the output from the first inception block as the second feature map. This map is passed through a convolutional layer with 112 filters of size 3×3 and a 2×2 maxpooling layer to down sample each of its spatial dimensions in half. The last feature map is the output from the second inception block which is fed to the concatenation operation as it is. Another fully connected layer with 192 hidden units is used, and the final output layer for the AFM contains 150 units and returns a map activated by a sigmoid to produce the AFM. To regularize the generator from overfitting on the training set, dropouts are introduced after each convolutional and fully connected component with the drop rates of 10% and 25% respectively.

The decoder: the backbone LSTM of the decoder consists of 256 hidden units. To compute the initial hidden state of the LSTM, a BLSTM of size 128 is used to process the context sequence to obtain a vector of size 256 (again the BLSTM produces two sequences of vectors of size 128 which are concatenated and averaged to get the size of 256). The attention module, which is a small neural network used to compute the weight coefficient $\alpha_{i,j}^{(t)}$ at each location (i, j) and at each time step t, is composed of one hidden layer with 128 units and activated by a sigmoid. In our experiments, we limit the output character from the decoder to be between U1780 and U17E9. The character which is not in this range is considered to be c_{unk}. Inspired by the work of [105], we adopt the beam search with length normalization technique to output the top k (we choose $k = 5$) predictions at each time step. The final text prediction is the sequence whose joined probability of each element is the maximum.

7.2.5. Training

The proposed network is trained from end to end in a supervised manner. The common approach is to train the generator and the decoder

jointly by minimizing the text prediction loss L_{TEXT} which is a cross-entropy loss computed by

$$L_{TEXT} = -\sum_t log\, p(c^{(t)} = \hat{c}^{(t)}) \qquad (7.16)$$

where $\hat{c}^{(t)}$ is the ground truth output character at time step t. We also want to introduce the GCM loss corresponding to the predicted GCM by the generator. By enforcing the generator to generate the GCM which contains both the identity and the spatial information of all glyphs in the input image, it is expected that this encoded context improves the recognition performance of the decoder. The GCM loss L_{GCM} is also a cross-entropy loss summing over all locations (i, j) in the GCM map computed by

$$L_{GCM} = -\sum_i \sum_j log\, p(g_{i,j} = \hat{g}_{i,j}) \qquad (7.17)$$

where $g_{i,j}$ is the predicted glyph and $\hat{g}_{i,j}$ is the ground truth glyph at position (i, j). The objective function of the network is therefore

$$L = L_{TEXT} + \lambda_s L_{GCM} \qquad (7.18)$$

where $\lambda_s \geq 0$ (we choose $\lambda_s = 1$ in our experiments) controls how predicting correct GCM affects the overall performance of the network.

The training is performed using the ADAM optimizer [102] per mini batch basis. To optimize the batch samples, prior to training, all sample image patches are sorted by their width so that the patches with similar dimension are put in the same batch. The images in each batch are upsampled to the size of the image with the maximum width to ensure that all images in the batch are of the same dimension. At the start of each epoch, the order of the batch is shuffled so that the network can learn to generalize well and to overfit less. After each batch, we accumulate gradients, and we update the parameters of the network only after multiple batches have been processed. This technique is equivalent to using a big batch size and proves to be useful in a situation where the computing resource is limited. In our experiments, we choose the batch size to be 8 and the number of batches for the parameters to be updated is 40. After every 200 iterations have been performed, we evaluate the

network on the validation set, and stop the training if the performance of the network does not improve after 5 epochs.

7.3. Segmenting into Groups of Sub-syllables and Merging their Transcriptions

Both the Word-Net and the SubSyl-Net can also be applied as segment-free by training the networks using whole text line images. However, there are some drawbacks as followings:

1) Each page of Khmer palm leaf manuscripts is very long which also produces long lines which are often curved or fluctuated. Therefore, in order for an image to contain a whole text line, parts of texts from neighboring lines might also appear in the image unless transformation techniques are applied to the image beforehand. This might also distort the texts in the image.
2) The number of samples in the training set would be significantly reduced since we use images of whole text lines instead of short words.
3) The computation of too long dependencies would also become very expensive.

It is therefore more beneficial that after separating the document page into individual text lines, the next task is to further segment them into smaller size image patches before feeding them to the text recognition system. The idea is that for each text line, a number of candidate horizontal locations are to be computed. Attributable to the non-cursiveness of Khmer writing, it is possible to look for small separating gaps between two neighboring handwritten characters. We reuse the SWT map calculated in the line segmentation stage (Chapter 5.2.1) to produce a horizontal projection histogram for each of the text lines. The histogram is smoothed multiple times using an average filter, and we then calculate the local minima of the smoothed histogram to obtain the candidate segmentation locations $X = (x_0, x_1, \dots, x_{N_{minima}-1})$ where N_{minima} is the number of local minima found in each line. Using these candidate locations, small image patches can be constructed following a number of criteria. Let's denote W_{cc} the most common width acquired

from the connected components extracted from the SWT map. This width is the median value of the sorted list of all connected component widths. An image patch is constructed by segmenting the text line between two candidates x_i and x_j if the width of the image patch stays between two hyper-parameters W_{MIN} and W_{MAX} i.e. $W_{MIN} < |x_j - x_i| < W_{MAX}$. In our experiment, we choose empirically $W_{MIN} = 2W_{cc}$ and $W_{MAX} = 8W_{cc}$.

Some image patches might not contain any foreground text. To detect and remove such image patches, we look at values between x_i and x_j in the smoothed histogram. We consider that the image patch contains foreground text if the number of zero values within this range is less than ¾ of $|x_j - x_i|$ otherwise the image is considered to be blank and is removed. By applying this process, we can obtain from a segmented text line, a sequence of image patches containing a group of glyphs (or a group of sub-syllables). Each image patch is to be fed to the text recognition system to return its text label as output. The text transcription of the entire line is therefore the concatenation of those short text labels.

7.4. Experiments and Results

7.4.1. Datasets

Similar to Trial-Net1 and Trial-Net2, the dataset used to train the Word-Net is generated from SleukRith set (see Chapter 6.1.2.1). For the network to recognize image patches composing groups of sub-syllables (SubSyl-Net), we train it with the newly created data called groups of sub-syllables also extracted from SleukRith set. Again, the data is divided into three subsets: the training set, the validation set, and the testing set. Each patch image in the data samples can be considered to contain synthetic texts since they are groups of sub-syllables which are not real Khmer words but still share some resemblance. We select n, the number of sub-syllables in each group, to be 3 and 4 due to the fact that these numbers of sub-syllables are the most common. Table 7.1 shows the number of image samples in each division from both the annotated word dataset and the newly created dataset of groups of sub-

syllables. We can also see that using the technique to create groups of sub-syllables instead of real words can augment the size of the dataset significantly (around tenfold).

Table 7.1: number of samples of the word dataset and the dataset of groups of sub-syllables

Type		Training Set	Validation set	Testing Set
Word Dataset		**15,432**	**813**	**7,764**
Sub-Syllables Dataset	**3-SG**	51,906	4,828	38,090
	4-SG	52,086	4,704	37,592
	Total SG	**103,992**	**9,532**	**75,682**

7.4.2. Evaluation Protocols

We use the same top k error rate measurement (as mentioned in Chapter 6.1.2.2) to evaluate the performance of the GCM generator of the Word-Net. To evaluate the performance of the GCM encoder-decoder of the Word-Net, Levenshtein distance D_L is used to compute the character error rate (CER) of each word as follows

$$CER = \frac{min\ (|Y_{gt}|, D_L(Y_{pred}, Y_{gt}))}{|Y_{gt}|} \tag{7.19}$$

where Y_{pred} and Y_{gt} are the predicted transcription and the ground truth transcription respectively, and $|Y_{gt}|$ represents the length of the ground truth transcription. According to this computation, the CER of each word is always between [0,1]. This also illustrates that the error rate is higher for the same amount of incorrectly predicted characters when the network performs on a shorter word image patch which makes sense since the importance of each character is stronger in short length transcriptions. The final CER is the average of each word CER in the test set. Word error rate (WER), which is the number of incorrectly predicted words over the total number of words, is also calculated for the evaluation.

For the SubSyl-Net, the evaluation is based on two criteria i.e. the generated GCM predicted by the generator and the text transcription

output by the decoder. A score $s \in [0,1]$ is calculated for each predicted GCM by

$$s = \frac{\sum_i \sum_j f(g_{i,j}, \hat{g}_{i,j})}{H_{GCM} * W_{GCM}} \quad (7.20)$$

where $f(g_{i,j}, \hat{g}_{i,j}) = 1$ if the predicted glyph $g_{i,j}$ matches the ground truth $\hat{g}_{i,j}$, otherwise it returns 0. The overall score is then computed as the mean of the scores of all predicted GCM over the testing set. To evaluate the text prediction of the network, we compute the Levenshtein distance between the output text from the decoder and the ground truth text. The character error rate (CER) can be obtained by summing the distances between all prediction and ground truth pairs and divide it by the sum of total number of characters in the ground truth text of all samples in the testing set. We also compute the word error rate (WER) which is the number of incorrectly predicted groups of sub-syllables over the total number of groups of sub-syllables.

The experimental evaluation for the SubSyl-Net is performed in three different scenarios to keep track of how each stage of the recognition pipeline affects the final prediction result.

- **Scenario 1:** we evaluate on the dataset of groups of sub-syllables, the overall flow of the proposed network.
- **Scenario 2:** we evaluate the network on image patches generated from the ground truth of line segmentation. The boundaries from the annotated glyph dataset are also used to split the ground truth text lines into smaller patches containing at most four sub-syllables. The CER is now computed as the Levenshtein distance between the predicted text line transcription (concatenation of all text labels in the image patch sequence) and the ground truth text line transcription which can be constructed as mentioned in Chapter 4.2.5.
- **Scenario 3:** we evaluate the performance of the network on the dataset of whole text lines. For each document page in the testing set, we segment it into lines and then into sequences of

image patches using the approach mentioned in Chapter 7.3. The CER is computed as in Scenario 2.

7.4.3. Hyper-parameter Tuning

The process of setting the hyper-parameters for deep learning models requires expertise and extensive trial and error. There are no simple and easy ways for hyper-parameter tuning. Some of the most common hyper-parameters include the number of layers, the number of units in each layer, the batch size, the number of filters and the filter size in CNNs …etc. Grid search is one popular technique where we set up a grid of hyper-parameters and train/test our model on each of the combinations. When using grid search, all possible combinations of the hyper-parameters are tried; therefore, the number of experiments increases exponentially with respect to the number of hyper-parameters in the model. In practice, this technique is very computationally expensive and time consuming.

In our experiments, to facilitate the tuning process, a manual search is performed instead where hyper-parameters are chosen one at a time. At the beginning of the process, each hyper-parameter is given an initial value, and after each trial, those values are updated empirically by different predefined intervals according to our observations. For example, after each experiment, the number of layers will be incremented or decremented by 1, the number of neurons in each layer will be modified by a factor of 2, and the learning rate will be changed by a factor of 10. During the fine-tuning experiments, the model is evaluated on a small validation set. We normally observe the changes on the results only after the first couple of epochs have been complete. If there is no significant improvement, we stop the experiment, decide on the best value for the current hyper-parameter so far, and move on to the next hyper-parameter.

The degree of importance of each hyper-parameter is also distingue. For instance, the performance of the Word-Net is not very sensitive to the hyper-parameter λ_w in Equation 7.10 (as illustrated by the results from Table 7.2). This shows the importance of the recognition loss L_1 compared to the GCM prediction loss L_2. Accordingly, for the SubSyl-

Net, we modify the balance between the two losses in Equation 7.18 (now denoted L_{TEXT} and L_{GCM} respectively) by introducing the hyper-parameter λ_s to be the coefficient of L_{GCM} alone. This way, the text recognition loss L_{TEXT} does not lose its importance by scaling according to λ_s.

7.4.4. Results and Discussion

Table 7.2 shows evaluation results of the Word-Net. Three experiments are conducted on the complete network after its two modules (the GCM generator and the GCM encoder-decoder) are pretrained separately to minimize its L_1 and L_2 respectively:

1) we do not do any finetuning
2) we finetune the complete network on L_{total} setting the hype-parameter λ_w to zero, i.e. L_1 has no effect on the total loss L_{total}
3) we finetune on L_{total} with $\lambda_w = 0.9$ (very strong influence of L_1)

By looking at the big difference between the top 1 and top 5 error rate of the generated GCM, it is illustrated that even though the GCM generator is sometimes not able to predict the correct glyph class as the most probable (top 1), in most of those cases, the probability of the correct glyph class is still high enough to be among the top 5. Fortunately, in the complete system, this glyph class probability distribution of the predicted GCM is passed directly to the GCM encoder-decoder which can be helpful for the generation of the final word transcription.

As seen in Table 7.2, finetuning the complete network by minimizing L_{total} improves the overall performance. Moreover, by enforcing the network to produce a good GCM (i.e. set a high value to λ_w), the error rates of the predicted word transcription decrease even more.

Table 7.2: Evaluation results of the Word-Net

	Error Rate of the GCM Generator (%)		**Error Rate of the Complete Pipeline (%)**	
	Top 1	**Top 5**	**CER**	**WER**
(1) No finetuning	12.42	0.25	4.43	13.49
(2) Finetune on L_{total} with $\lambda = 0$	12.81	0.24	3.88	12.11
(3) Finetune on L_{total} with $\lambda = 0.9$	12.21	0.23	3.80	11.81

The evaluation results of the three scenarios for the SubSyl-Net are shown in Table 7.3. On the newly constructed dataset of groups of sub-syllables, the proposed feature generator obtains a sufficiently high score in producing the output GCM. We observe that although the predicted GCM contains some noise and does not match perfectly cell by cell with the ground truth, it is still capable of storing the useful information related to the spatial position, size, and identity of each glyph in the input image. To highlight this and also the effectiveness of the GCM on the attention mechanism of the proposed decoder, Figure 7.6 illustrates some sample outputs and the attention map at each time step t ($\alpha_{i,j}^{(t)}$) corresponding to each predicted character. According to this result, it is shown that the generated GCM from the generator serves as a blueprint which contains candidate regions for the decoder to attend to. As expected, by enforcing the generator to learn to produce the ground truth GCM, the spatial information of each glyph annotated in the GCM is helpful in enhancing the attention mechanism as well as improving the predicting capability of the decoder. We measure the performance of the overall network by computing the CER and the WER of the predicted text labels. According to these results on the new dataset, the proposed network is able to achieve a low CER and, as expected, a slightly higher WER since the text labels in the dataset are often long.

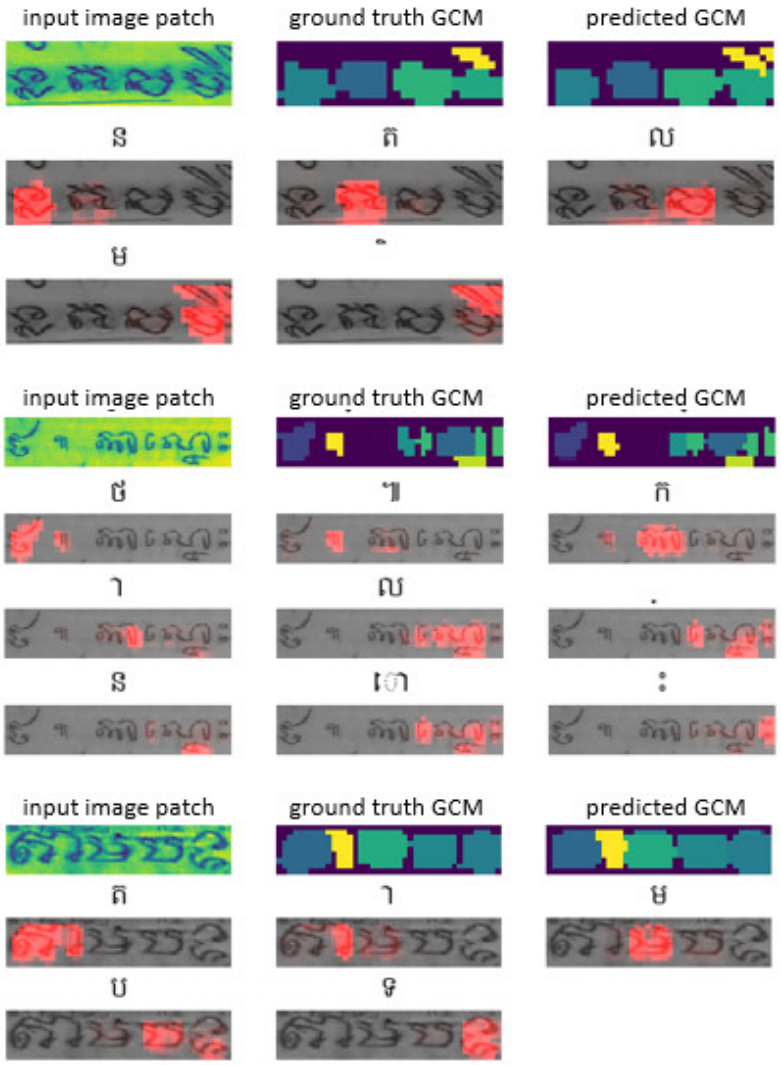

Figure 7.6: Sample results from the SubSyl-Net showing the predicted GCM and the attention map at each time step (the region highlighted in red is where the decoder attends to)

In Scenario 3, we also evaluate the performance of the proposed system on the input images obtained from segmenting the whole document page into lines which are then split into short patches. In this scenario, the network does not perform as well. However, according to the result from Scenario 2 which shows a low CER, it can be assumed that this problem is caused by the faults at the segmentation stage rather than at

the recognition stage. The line segmentation and image patch extraction approach could therefore be improved.

Table 7.3: Evaluation results of the SubSyl-Net

Scenario	GCM Score (%)	CER (%)	WER (%)
1	87.12	6.16	26.23
2	-	7.81	-
3	-	35.30	-

We also perform a comparison with the results obtained from the Word-Net (which is trained on the word dataset). Compared to the Word-Net, the number of parameters in the SubSyl-Net is significantly reduced as illustrated in Table 7.4. According to the results shown in the table, although the Word-Net performs rather well on annotated word dataset, it does not generalize to sub-syllable dataset which is much bigger. This shows the overfitting problem of the Word-Net. In contrast, the SubSyl-Net which is trained on the expanded dataset using the proposed data augmentation technique generalizes well to all datasets. The SubSyl-Net produces a satisfactory result on the word dataset which it has never seen before (we do not train the SubSyl-Net on the word dataset). Using groups of sub-syllables as a technique to augment the word image patch sample proves to be efficient. Along with the optimized architecture of the SubSyl-Net, we are able to solve the overfitting problem of the Word-Net as well.

Table 7.4: Comparison between Word-Net and SubSyl-Net

Network	Number of parameters (Millions)	Dataset	CER (%)	WER (%)
Word-Net	9.18	Words	3.80	11.81
		Groups of Sub-syllables	49.62	96.42
SubSyl-Net	2.37	Words	6.88	19.33
		Groups of Sub-syllables	6.16	26.23

8. Conclusions

8.1. Summary

This thesis presents a number of contributions in the field of DIA especially on the application of handwritten recognition on old and degraded historical documents. The main research work in this thesis aims at designing and implementing an efficient text recognition system which is expected to help widen the accessibility of the valuable contents written of palm leaf manuscripts to the public.

In order to achieve the goal of developing such handwritten recognition system, digital corpuses and datasets of Khmer palm leaf documents are of fundamental interest for benchmarking existing recognition systems as well as for the training of data-driven recognition methods and the experimental evaluation of their performance. In this dissertation, we introduce SleukRith Set, the first dataset constructed on digital images of Khmer palm leaf manuscripts from our own digitization campaign and also from existing digital content from various establishments. We select 657 manuscript pages, and a tool has been developed to perform the annotation task on those pages to create three types of data: isolated character dataset, annotated word dataset, and line segmentation ground truth. In future work, the next version of the dataset is likely to include an increased number of pages so that it can be used as training data for even more sophisticated systems. The dataset and also the annotation tool are made publicly available at github.com/donavaly/SleukRith-Set.

Next, a comprehensive experimental study for the binarization task which is one the principal preprocessing steps in a DIA system is presented. With the special characteristics and challenges possessed by the palm leaf manuscript collections such as degradations, artifacts, different kinds of noises the performances of the binarization methods in the literature on this type of documents are not satisfactory. By observing the output binary images obtained after applying the binarization methods, many broken and unrecognizable characters/glyphs and noises still occur. This illustrates a great challenge in binarizing digitized palm leaf manuscripts.

According to the difficulty in obtaining satisfactory results from the binarization step, a binarization-free line segmentation scheme that works directly on grayscale images is proposed for Khmer palm leaf manuscripts. First, we apply a stroke width transform (SWT) on the edge map of the image to extract connected components. The proposed approach uses piece-wise project profiles to detect line number and to set up initial positions of the centroids or text line mid-points. Those centroids are then moved adaptively to form correct lines by applying a clustering algorithm called competitive learning. Finally, we construct text line boundaries using a path finding technique. The evaluation experiments on digitized pages from SleukRith set as well as on additional datasets constructed from documents of various Indonesian scripts demonstrate that the proposed approach produces a very promising outcome compared to other existing methods in the literature. The method performs well on historical palm leaf documents whose lines are often skewed or even fluctuated. It can also deal with discontinuity of text lines caused by holes made for document page binding.

After dealing with the preprocessing step which is to segment the document page into separating text lines, we conduct the first attempt in using neural networks to solve the text recognition problem for Khmer palm leaf manuscripts. To study their feasibility and performance, different types of ANN architectures are used for the recognition of character and word image patches extracted from Khmer ancient palm leaf documents. For the task of classifying isolated characters, we present four network architectures: purely CNN based, RNN on sequence of one-pixel columns, RNN on both column and row sequences, and finally a combination which is both convolutional and recurrent. The results show that both CNN and RNN based architectures perform well enough on this task individually; however, combining both types of architectures proves to be better and more powerful. For the text word recognition task, SleukRith Set provides character annotation which can be used as alignment between character codes in the text transcription and the character positions in the text images. To incorporate this information, we propose approaches whose objective is

to output a glyph-class map (GCM) for each input word image using both one-dimensional and two-dimensional recurrent neural networks. For this task, it is illustrated in the evaluation results that the latter network performs better. The predicted GCM can be used as input to produce the final text transcription of the word.

Since utilizing neural networks to solve Khmer handwritten text recognition problem in our trial experimentations is very favorable, we present a robust system called Word-Net whose architecture uses and combines both the convolutional and recurrent modules to take as input an image patch containing one Khmer word and to output its corresponding text transcription. The proposed system takes advantages also from the GCM constructed using the glyph annotation which contains information about the structure, position, and identity of each glyph in the word image to be recognized. Two main modules, the GCM generator and the GCM encoder-decoder are developed to generate the GCM which is to be encoded into a context vector and also local contexts representing the input word image before being decoded into the final transcription. Our approach shows promising results evaluated on the word dataset extracted from SleukRith set.

However, due to the limited size of the annotated word dataset used to train the Word-Net, it does not generalize well i.e. it performs poorly on documents containing words that do not appear in the word dataset (for example, the many manuscript collections written in Pali script where words cannot be easily annotated). To overcome this challenge, we construct a new significantly larger dataset composing of groups of sub-syllables. Synthetic words which resemble real Khmer words can be formed by assembling adjacent annotated sub-syllables. To accommodate this new dataset, we also propose a new text recognition system (denoted SubSyl-Net) designed to take into account the limited computing resource which is very crucial in dealing with large size datasets. Similar to the Word-Net, the newly proposed network uses a generator to extracts a GCM and also an additional feature map so that these encoded features can be decoded by an attention-based decoder to output a corresponding text label of the input image patch. The experiments conducted on the new dataset show that, compared to the

Word-Net, the SubSyl-Net performs and generalizes promisingly well on all palm leaf documents.

8.2. Impacts

Palm leaf manuscripts have become an integral part of Cambodian culture and are of great importance especially due to the scarcity of scripters who are capable of producing new manuscripts as well as the fact that aging manuscripts are facing destruction. The salvation efforts have been made, and the digitized material of Khmer palm leaf documents are becoming available in increasingly large quantity. Making the content of those digital copies easily accessible to the public is a great challenge. This research is the first step in solving this problem.

The outcomes from this research work open ways to development of numerous practical applications. For instance, keyword indexing will be able to be made possible to enhance the in-text search capability so that a relevant document can be retrieved through word spotting scheme with respect to a query of the user. The proposed text recognition system can also be used as a tool to extract and analyze the handwritten texts from the document images which are very useful as inputs to other popular applications such as machine translation and text to speech. These applications will make Khmer palm leaf manuscripts accessible to an even bigger audience.

Historical documents written in languages with similar writing nature (for example, Thai, Laos as well as ancient scriptures such as Pali and Sanskrit) also benefit from this research work.

8.3. Future Work

This dissertation has tackled a number of different tasks in DIA on historical palm leaf manuscripts. There are still however several possibilities as an extension for improvement and for future research.

Segmentation using Deep Learning: The proposed text recognition system works sufficiently well to recognize short image patches. To apply this work however to the whole document page, a method to efficiently extract text lines as well as to segment those lines into

smaller patches is still needed. In this dissertation, a line segmentation approach following a bottom up scheme is proposed. This approach proves good performance since grouping text components into lines does not necessarily require language context. Word segmentation however is a much harder problem. As illustrated by the result form Scenario 3 in Table 7.3, separating lines into short patches just by simply using the gaps between adjacent glyphs (described in detail in Chapter 7.3) does not produce a satisfactory outcome. Some common erroneous cases include separation of characters with multiple parts and patches containing noises or only background with no text. A different approach that we can pursue is to apply deep neural network models for word segmentation since models such as RNNs are able to take advantage of sequential information between neighboring text components in the line. Global processing of both line and word segmentation in a single task can also be another direction to tackle. For these deep learning approaches, significantly larger datasets of line and word segmentation are very crucial.

Postprocessing: For an even further improvement of the text recognition system, postprocessing steps are needed. Using the contextual information of the natural language, additional errors can be detected and corrected. Both error detection and error correction models are to be explored. One of the solutions is to adopt and integrate language models which construct probability distribution over all strings in the language. These models can be implemented in different levels of context i.e. characters, words, or even short phrases. To train such models, SleukRith Set will be utilized. In addition, more annotated data such as syntactic structure and grammatical form are to be constructed if necessary.

Complete End-to-End and Multi-task System: As mentioned in previous chapters, solving DIA problems, developing a handwritten text recognition system in particular, does not consist of a single task but often a sequence of steps composing a complete pipeline. While it is feasible to propose and improve unique models to solve each task individually, another direction might be to develop an end-to-end system that is able to perform multiple tasks simultaneously or

sequentially but in a single process flow. Usually errors occurring at the first stages of the workflow significantly affect the outcomes in the later stages. By chaining all tasks together, the accumulation of errors from those related tasks might be useful in improving the overall performance of the complete system. Using deep learning approaches to model such sequential end-to-end workflow is expected to be practical and efficient due to the backward propagation where errors from the later layers which correspond as well to the later tasks can be fed back to fix the faults made by the first layers which belong to the task at the beginning stage of the system. This end-to-end learning technique is relatively new and is still an ongoing research. Some recent examples include the Start-Follow-Read model [108, 109] which is composed of modules to perform both segmentation and recognition. The recognition errors from the last module are not only used to correct the recognition module itself but also to improve the segmentation module.

Transfer Learning: Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. Given the vast computing and time resources required to develop neural network models, instead of building models for certain tasks from scratch it is beneficial to use existing pretrained networks of other related tasks. Handwritten text recognition on historical documents is a universal problem. Numerous datasets from different language sources along with their DIA systems have been constructed, implemented, and made available. However, the questions of what to be transferred (i.e. the common useful information which can be extracted from those sources) and how the transfer is performed for Khmer palm leaf documents might be challenging and in need of further study.

References

[1] D. Valy, M. Verleysen, S. Chhun and J.-C. Burie, "A New Khmer Palm Leaf Manuscript Dataset for Document Analysis and Recognition: SleukRith Set," in *4th International Workshop on Historical Document Imaging and Processing*, 2017.

[2] M. W. Kesiman, D. Valy, J.-C. Burie, E. Paulus, I. M. G. Sunarya, S. Hadi, K. H. Sok and J.-M. Ogier, "Southeast Asian palm leaf manuscript images: a review of handwritten text line segmentation methods and new challenges," *Journal of Electronic Imaging,* vol. 26, no. 1, p. 011011, 2016.

[3] M. W. Kesiman, D. Valy, J.-C. Burie, E. Paulus, M. Suryani, S. Hadi, M. Verleysen, S. Chhun and J.-M. Ogier, "Benchmarking of Document Image Analysis Tasks for Palm Leaf Manuscripts from Southeast Asia," *Journal of Imaging,* vol. 4, no. 2, p. 43, 2018.

[4] D. Valy, M. Verleysen and K. H. Sok, "Line segmentation approach for ancient palm leaf manuscripts using competitive learning algorithm," in *15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016.

[5] D. Valy, M. Verleysen and K. H. Sok, "Line segmentation for grayscale text images of khmer palm leaf manuscripts," in *Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, 2017.

[6] D. Valy, M. Verleysen, S. Chhun and J.-C. Burie, "Character and Text Recognition of Khmer Historical Palm Leaf Manuscripts," in *16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2018.

[7] D. Valy, M. Verleysen and S. Chhun, "Text Recognition on Khmer Historical Documents using Glyph Class Map

Generation with Encoder-Decoder Model," in *Proceedings of ICPRAM*, 2019.

[8] J.-C. Burie, M. Coustaty, S. Hadi, M. W. Kesiman, J.-M. Ogier, E. Paulus, K. Sok, I. M. G. Sunarya and D. Valy, "ICFHR2016 competition on the analysis of handwritten text in images of balinese palm leaf manuscripts," in *15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016.

[9] M. W. Kesiman, D. Valy, J.-C. Burie, E. Paulus, M. Suryani, S. Hadi, M. Verleysen, S. Chhun and J.-M. Ogier, "ICFHR 2018 Competition On Document Image Analysis Tasks for Southeast Asian Palm Leaf Manuscripts," in *16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2018.

[10] D. U. Kumar, G. Sreekumar and U. Athvankar, "Traditional writing system in southern India—palm leaf manuscripts," *Design Thoughts,* pp. 2-7, 2009.

[11] O. P. Agrawal, "Conservation of Manuscripts and Paintings of South-east Asia," *London: Butterworths & Co Ltd.,* 1984.

[12] J. M. Jacob and D. A. Smyth, Cambodian Linguistics, Literature and History: Collected Articles, Routledge, 1993.

[13] C. Sak-Humphry, "The Syntax of Nouns and Noun Phrases in Dated Pre-Angkorian Inscriptions," *Mon Khmer Studies,* vol. 22, p. 1–26, 1993.

[14] N. B. Rais, M. S. Hanif and I. A. Taj, "Adaptive thresholding technique for document image analysis," in *8th International Multitopic Conference*, 2004.

[15] K. Ntirogiannis, B. Gatos and I. Pratikakis, "An objective evaluation methodology for document image binarization

techniques," in *The Eighth IAPR International Workshop on Document Analysis Systems*, 2008.

[16] R. Chamchong, C. C. Fung and K. W. Wong, "Comparing binarisation techniques for the processing of ancient manuscripts," in *Entertainment Computing Symposium*, 2010.

[17] J. He, Q. D. M. Do, A. C. Downton and J. Kim, "A comparison of binarization methods for historical archive documents," in *Eighth International Conference on Document Analysis and Recognition*, 2005.

[18] M. R. Gupta, N. P. Jacobson and E. K. Garcia, "OCR binarization and image pre-processing for searching historical documents," *Pattern Recognition,* vol. 40, no. 2, pp. 389-397, 2007.

[19] M.-L. Feng and Y.-P. Tan, "Contrast adaptive binarization of low quality document images," *IEICE Electronics Express,* vol. 1, no. 16, pp. 501-506, 2004.

[20] K. Khurshid, I. Siddiqi, C. Faure and N. Vincent, "Comparison of Niblack inspired Binarization methods for ancient documents," *Document Recognition and Retrieval XVI,* vol. 7247, p. 72470U, 2009.

[21] J. Sauvola and M. Pietikäinen, "Adaptive document image binarization," *Pattern recognition,* vol. 33, no. 2, pp. 225-236, 2000.

[22] C. Wolf, J.-M. Jolion and F. Chassaing, "Text localization, enhancement and binarization in multimedia documents," in *Object recognition supported by user interaction for service robots*, 2002.

[23] N. Tripathy and U. Pal, "Handwriting segmentation of unconstrained Oriya text," *Sadhana,* vol. 31, no. 6, pp. 755-769, 2006.

[24] R. Ptak, B. Żygadło and O. Unold, "Projection–Based Text Line Segmentation with a Variable Threshold," *International Journal of Applied Mathematics and Computer Science,* vol. 27, no. 1, pp. 195-206, 2017.

[25] M. Arivazhagan, H. Srinivasan and S. Srihari, "A statistical approach to line segmentation in handwritten documents," *Document Recognition and Retrieval XIV,* vol. 6500, p. 65000T, 2007.

[26] R. Chamchong and C. C. Fung, "Text line extraction using adaptive partial projection for palm leaf manuscripts from Thailand," in *International Conference on Frontiers in Handwriting Recognition*, 2012.

[27] A. Nicolaou and B. Gatos, "Handwritten text line segmentation by shredding text into its lines," in *10th International Conference on Document Analysis and Recognition*, 2009.

[28] G. Peng, P. Yu, H. Li and L. He, "Text line segmentation using Viterbi algorithm for the palm leaf manuscripts of Dai," in *International Conference on Audio, Language and Image Processing*, 2016.

[29] X. Zhang and C. L. Tan, "Text line segmentation for handwritten documents using constrained seam carving," in *14th International Conference on Frontiers in Handwriting Recognition*, 2014.

[30] O. Surinta, M. Holtkamp, F. Karabaa, J.-P. V. Oosten, L. Schomaker and M. Wiering, "A* path planning for line segmentation of handwritten documents," in *14th International Conference on Frontiers in Handwriting Recognition*, 2014.

[31] Y. Li, Y. Zheng, D. Doermann and S. Jaeger, "Script-independent text line segmentation in freestyle handwritten documents," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 30, no. 8, pp. 1313-1329, 2008.

[32] G. Louloudis, B. Gatos and C. Halatsis, "Text line detection in unconstrained handwritten documents using a block-based hough transform approach," in *Ninth International Conference on Document Analysis and Recognition*, 2007.

[33] D. Brodić and Z. N. Milivojević, "Text line segmentation with the parametric water flow algorithm," *Information Technology And Control,* vol. 45, no. 1, pp. 52-61, 2016.

[34] I. Bar-Yosef, N. Hagbi, K. Kedem and I. Dinstein, "Line segmentation for degraded handwritten historical documents," in *10th International Conference on Document Analysis and Recognition*, 2009.

[35] A. Garz, A. Fischer, H. Bunke and R. Ingold, "A binarization-free clustering approach to segment curved text lines in historical manuscripts," in *12th International Conference on Document Analysis and Recognition*, 2013.

[36] N. Arvanitopoulos and S. Süsstrunk, "Seam carving for text line extraction on color and grayscale historical manuscripts," in *14th International Conference on Frontiers in Handwriting Recognition*, 2014.

[37] M. W. A. Kesiman, J.-C. Burie and J.-M. Ogier, "A new scheme for text line and character segmentation from gray scale images of palm leaf manuscript," in *15th International Conference on Frontiers in Handwriting Recognition*, 2016.

[38] A. Aggarwal, K. Singh and K. Singh, "Use of gradient technique for extracting features from handwritten gurmukhi characters

and numerals," *Procedia Computer Science,* vol. 46, pp. 1716-1723, 2015.

[39] M. Blumenstein, B. Verma and H. Basli, "A novel feature extraction technique for the recognition of segmented handwritten characters," in *Seventh International Conference on Document Analysis and Recognition*, 2003.

[40] Z. Jin, K. Qi, Y. Zhou, K. Chen, J. Chen and H. Guan, "Ssift: An improved sift descriptor for chinese character recognition in complex images," in *International Symposium on Computer Network and Multimedia Technology*, 2009.

[41] M. Rani and Y. K. Meena, "An efficient feature extraction method for handwritten character recognition," in *International Conference on Swarm, Evolutionary, and Memetic Computing*, 2011.

[42] Ø. D. Trier, A. K. Jain and T. Taxt, "Feature extraction methods for character recognition-a survey," *Pattern recognition,* vol. 29, no. 4, pp. 641-662, 1996.

[43] S. Impedovo, L. Ottaviano and S. Occhinegro, "Optical character recognition—a survey," *International Journal of Pattern Recognition and Artificial Intelligence,* vol. 5, no. 01n02, pp. 1-24, 1991.

[44] U. Pal and B. B. Chaudhuri, "Indian script character recognition: a survey," *Pattern Recognition,* vol. 37, no. 9, pp. 1887-1899, 2004.

[45] M. Glauberman, "Character recognition for business machines," *Electronics,* vol. 29, no. 2, pp. 132-136, 1956.

[46] P. Bao-Chang, W. Si-Chang and Y. Guang-Yi, "Amethod of Recognizing handprinted characters," *Computer Recognition and Human Production of Handwriting,* pp. 37-60, 1989.

[47] M. Shridhar and A. Badreldin, "Recognition of isolated and connected handwritten numerals," in *IEEE International Conference on Systems, Man and Cybernetics*, 1984.

[48] R. A. Kirsch, "Computer determination of the constituent structure of biological images," *Computers and biomedical research,* vol. 4, no. 3, pp. 315-328, 1971.

[49] G. H. Granlund, "Fourier preprocessing for hand print character recognition," *IEEE transactions on computers,* vol. 100, no. 2, pp. 195-201, 1972.

[50] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE transactions on information theory,* vol. 8, no. 2, pp. 179-187, 1962.

[51] M. Bokser, "Omnidocument technologies," *Proceedings of the IEEE,* vol. 80, no. 7, pp. 1066-1078, 1992.

[52] M. Z. Hossain, A. Amin and H. Yan, "Rapid feature extraction for optical character recognition," in *arXiv preprint arXiv:1206.0238*, 2012.

[53] M. W. A. Kesiman, S. Prum, J.-C. Burie and J.-M. Ogier, "Study on feature extraction methods for character recognition of Balinese script on palm leaf manuscript images," in *23rd International Conference on Pattern Recognition*, 2016.

[54] S. Kumar, "Neighborhood Pixels Weights-A New Feature Extractor," *International Journal of Computer Theory and Engineering,* vol. 2, no. 1, p. 69, 2010.

[55] C. M. Bishop, Neural networks for pattern recognition, Oxford university press, 1995.

[56] S. Haykin, Neural networks: a comprehensive foundation, 2nd ed., Prentice Hall PTR, 2004.

[57] A. Karpathy, Cs231n convolutional neural networks for visual recognition, 2019.

[58] A. L. Maas, A. Y. Hannun and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," *Proc. icml,* vol. 30, no. 1, p. 3, 2013.

[59] D. E. Rumelhart, G. E. Hinton and W. Ronald J., "Learning representations by back-propagating errors," *Cognitive modeling,* vol. 5, no. 3, p. 1, 1988.

[60] I. Goodfellow, Y. Bengio and A. Courville, Deep learning, MIT press, 2016.

[61] Y. LeCun, "Generalization and network design strategies," *Connectionism in perspective,* vol. 19, 1989.

[62] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012.

[63] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *arXiv preprint arXiv:1409.1556*, 2014.

[64] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *IEEE conference on computer vision and pattern recognition*, 2016.

[65] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[66] C. Olah, "Understanding LSTM Networks," 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[67] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, 2000.

[68] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *arXiv preprint arXiv:1406.1078*, 2014.

[69] K. Yao, T. Cohn, K. Vylomova, K. Duh and C. Dyer, "Depth-gated LSTM," in *arXiv preprint arXiv:1508.03790*, 2015.

[70] J. Koutnik, K. Greff, F. Gomez and J. Schmidhuber, "A clockwork RNN," in *arXiv preprint arXiv:1402.3511*, 2014.

[71] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE transactions on neural networks and learning systems,* vol. 28, no. 10, pp. 2222-2232, 2016.

[72] R. Jozefowicz, W. Zaremba and I. Sutskever, "An empirical exploration of recurrent network architectures," in *International Conference on Machine Learning*, 2015.

[73] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing,* vol. 45, no. 11, pp. 2673-2681, 1997.

[74] A. Graves, S. Fernández and J. Schmidhuber, "Multi-dimensional recurrent neural networks," in *International conference on artificial neural networks*, 2007.

[75] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

[76] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine,* vol. 29, no. 6, pp. 141-142, 2012.

[77] G. M. d. B. Wenniger, L. Schomaker and A. Way, "No Padding Please: Efficient Neural Handwriting Recognition," in *arXiv preprint arXiv:1902.11208*, 2019.

[78] U.-V. Marti and H. Bunke, "Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system," *Hidden Markov models: applications in computer vision,* pp. 65-90, 2001.

[79] H. Bunke, S. Bengio and A. Vinciarelli, "Offline recognition of unconstrained handwritten texts using HMMs and statistical language models," *IEEE transactions on Pattern analysis and Machine intelligence,* vol. 26, no. 6, pp. 709-720, 2004.

[80] J. A. Sanchez, V. Romero, A. H. Toselli and E. Vidal, "ICFHR2016 competition on handwritten text recognition on the READ dataset," in *15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016.

[81] H. Ding, K. Chen, Y. Yuan, M. Cai, L. Sun, S. Liang and Q. Huo, "A compact CNN-DBLSTM based character model for offline handwriting recognition with Tucker decomposition," in *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[82] T. Bluche, J. Louradour and R. Messina, "Scan, attend and read: End-to-end handwritten paragraph recognition with mdlstm attention," in *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[83] D. Castro, B. L. Bezerra and M. Valença, "Boosting the deep multidimensional long-short-term memory network for handwritten recognition systems," in *16th International*

Conference on Frontiers in Handwriting Recognition (ICFHR), 2018.

[84] W. Wang, J. Zhang, J. Du, Z.-R. Wang and Y. Zhu, "DenseRAN for Offline Handwritten Chinese Character Recognition," in *16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2018.

[85] N. T. Ly, C. T. Nguyen and M. Nakagawa, "Training an End-to-End Model for Offline Handwritten Japanese Text Recognition by Generated Synthetic Patterns," in *16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2018.

[86] C. K. Nguyen, C. T. Nguyen and N. Masaki, "Tens of Thousands of Nom Character Recognition by Deep Convolution Neural Networks," in *4th International Workshop on Historical Document Imaging and Processing*, 2017.

[87] Y.-C. Wu, F. Yin, Z. Chen and C.-L. Liu, "Handwritten Chinese Text Recognition Using Separable Multi-Dimensional Recurrent Neural Network," in *14th IAPR International Conference on Document Analysis and Recognition*, 2017.

[88] A. Lawgali, "A survey on arabic character recognition," *International Journal of Signal Processing, Image Processing and Pattern Recognition,* vol. 8, no. 2, pp. 401-426, 2015.

[89] C. Clausner, A. Antonacopoulos, N. Mcgregor and D. Wilson-Nunn, "ICFHR 2018 Competition on Recognition of Historical Arabic Scientific Manuscripts–RASM2018," in *16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2018.

[90] A. Graves, S. Fernández, F. Gomez and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented

sequence data with recurrent neural networks," in *23rd international conference on Machine learning*, 2006.

[91] A. Telea, "An image inpainting technique based on the fast marching method," *Journal of graphics tools,* vol. 9, no. 1, pp. 23-34, 2004.

[92] M. W. A. Kesiman, J.-C. Burie, G. N. M. A. Wibawantara, I. M. G. Sunarya and J.-M. Ogier, "AMADI_LontarSet: The First Handwritten Balinese Palm Leaf Manuscripts Dataset," in *15th International Conference on Frontiers in Handwriting Recognition*, 2016.

[93] M. Suryani, E. Paulus, S. Hadi, U. A. Darsa and J.-C. Burie, "The Handwritten Sundanese Palm Leaf Manuscript Dataset From 15th Century," 2017.

[94] B. Gatos, K. Ntirogiannis and I. Pratikakis, "DIBCO 2009: document image binarization contest," *International Journal on Document Analysis and Recognition,* vol. 14, no. 1, pp. 35-44, 2011.

[95] I. Pratikakis, B. Gatos and K. Ntirogiannis, "ICDAR 2013 document image binarization contest (DIBCO 2013)," in *12th International Conference on Document Analysis and Recognition*, 2013.

[96] N. R. Howe, "Document binarization with automatic parameter tuning," *International Journal on Document Analysis and Recognition,* vol. 16, no. 3, pp. 247-258, 2013.

[97] M. W. A. Kesiman, S. Prum, J.-C. Burie and J.-M. Ogier, "An initial study on the construction of ground truth binarized images of ancient palm leaf manuscripts," in *13th International Conference on Document Analysis and Recognition*, 2015.

[98] E. Saund, J. Lin and P. Sarkar, "Pixlabeler: User interface for pixel-level labeling of elements in document images," in *10th*

International Conference on Document Analysis and Recognition, 2009.

[99] J. Canny, "A computational approach to edge detection," in *Readings in computer vision*, 1987.

[100] B. Epshtein, E. Ofek and Y. Wexler, "Detecting text in natural scenes with stroke width transform," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

[101] N. Stamatopoulos, B. Gatos, G. Louloudis, U. Pal and A. Alaei, "ICDAR 2013 handwriting segmentation contest," in *12th International Conference on Document Analysis and Recognition*, 2013.

[102] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *arXiv preprint arXiv:1412.6980*, 2014.

[103] M. W. A. Kesiman, J.-C. Burie and J.-M. Ogier, "A Complete Scheme Of Spatially Categorized Glyph Recognition For The Transliteration Of Balinese Palm Leaf Manuscripts," in *14th IAPR International Conference on Document Analysis and Recognition*, 2017.

[104] D. Bahdanau, K. Cho and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *arXiv preprint arXiv:1409.0473*, 2014.

[105] Y. Wu, M. Schuster, Z. Chen, Q. V. Le and M. Norouzi, "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," in *arXiv preprint arXiv:1609.08144*, 2016.

[106] J. Puigcerver, "Are multidimensional recurrent layers really necessary for handwritten text recognition?," in *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[107] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going Deeper with Convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[108] C. Wigington, C. Tensmeyer, B. Davis, W. Barrett, B. Price and S. Cohen, "Start, follow, read: End-to-end full-page handwriting recognition," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[109] B. Moysset, C. Kermorvant and C. Wolf, "Full-page text recognition: Learning where to start and when to stop," in *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

Appendix A

Different categories of Khmer symbols are listed as follows:

List of consonants

Consonant	Sub-consonant	Name[8]
ក	◌្ក	KA
ខ	◌្ខ	KHA
គ	◌្គ	KO
ឃ	◌្ឃ	KHO
ង	◌្ង	NGO
ច	◌្ច	CHA
ឆ	◌្ឆ	CHHA
ជ	◌្ជ	CHO
ឈ	◌្ឈ	CHHO
ញ	◌្ញ	NHO
ដ	◌្ដ	DA
ឋ	◌្ឋ	DHA
ឌ	◌្ឌ	DO
ឍ	◌្ឍ	DHO
ណ	◌្ណ	NA
ត	◌្ត	TA
ថ	◌្ថ	THA

Consonant	Sub-consonant	Name
ទ	◌្ទ	TO
ធ	◌្ធ	THO
ន	◌្ន	NO
ប	◌្ប	BA
ផ	◌្ផ	PHA
ព	◌្ព	PO
ភ	◌្ភ	PHO
ម	◌្ម	MO
យ	◌្យ	YO
រ	◌្រ	RO
ល	◌្ល	LO
វ	◌្វ	VO
ស	◌្ស	SA
ហ	◌្ហ	HA
ឡ		LA
អ	◌្អ	AA

[8] Name of each symbol written in Latin which is used to easily identify the symbol in this dissertation (this name does not represent the true transliteration of the symbol)

List of dependent vowels

Vowel	Example	Name
◌ា	កា	A
◌ិ	កិ	I
◌ី	កី	EI
◌ឹ	កឹ	E
◌ឺ	កឺ	EU
◌ុ	កុ	O
◌ូ	កូ	AU
◌ួ	កួ	UO

Consonant	Sub-consonant	Name
◌ើ	កើ	AEU
◌ឿ	កឿ	OEU
◌ៀ	កៀ	IEU
◌េ	កេ	EE
◌ែ	កែ	È
◌ៃ	កៃ	AI
◌ោ	កោ	AO
◌ៅ	កៅ	AOV

List of independent vowels

Vowel	Example	Name
ឥ	ឥណ្ឌា	Ind-I
ឦ	ឦសូរ	Ind-EI
ឧ	ឧក	Ind-U
ឩ	ឩកា	Ind-OU
ឪ	ឪពុក	Ind-EUV
ឫ	ឫក	Ind-RE

Consonant	Sub-consonant	Name
ឬ	ឬក៏	Ind-REU
ឭ	រឭក	Ind-LE
ឮ	ឮឈាន់	Ind-LEU
ឯ	ឯណា	Ind-È
ឰ	ឰដ៏	Ind-AI
ឱ	ឱកាស	Ind-AO

List of diacritics

Diacritic	Example	Description[9]
◌ំ	អ៊ុំទូក	It nasalizes the inherent or dependent vowel, with the addition of /m/; long vowels are also shortened.
◌ះ	សេះ	It modifies and adds final aspiration /h/ to the inherent or dependent vowel.

[9] https://en.wikipedia.org/wiki/Khmer_script#Diacritics

◌ះ	ស្រៈ	It is written after a consonant to indicate that it is to be followed by a short vowel and a glottal stop
◌ំ	ញ៉ាម	It is written above a consonant, used to convert some o-series consonants to a-series.
◌ិ	ប៊ិច	It is written above a consonant, used to convert some a-series consonants to o-series.
◌៎	កក់	It is written over the last consonant of a syllable, indicating shortening of certain vowels.
◌៍	ធម៌	It occurs in Sanskrit loanwords and originally represents an /r/ sound. Now, in most cases, the consonant above which it appears, and the diacritic itself, are unpronounced.
◌៑	សៅរ៑	It is written over a final consonant to indicate that it is unpronounced.
◌៓	ចាំ៓	It is used in writing to indicate the rising intonation of an exclamation or interjection.
◌៝	ក៏	It is used in a few words to show that a consonant with no dependent vowel is to be pronounced with its inherent vowel, rather than as a final consonant.
◌ៈ	ច័ន្ទ	It is used in some Sanskrit and Pali loanwords. It is written above a consonant to indicate that the syllable contains a particular short vowel.

List of punctuations

Punctuation	Description[10]
។	It is used as a period
៕	It is written after a consonant to indicate that it is to be followed by a short vowel and a glottal stop.
។ល។	It is equivalent to etc.
ៗ	Duplication sign. It indicates that the preceding word or phrase is to be repeated, a common feature in Khmer syntax.
ᨀ	A period used at the end of poetic or religious texts.
ᩝ	A symbol used at the start of poetic or religious texts.
៖	Used similarly to a colon.

[10] https://en.wikipedia.org/wiki/Khmer_script#Spacing and punctuation

List of numerals

Khmer	០	១	២	៣	៤	៥	៦	៧	៨	៩
Arabic	0	1	2	3	4	5	6	7	8	9

Appendix B

Khmer Unicode[11] (the Unicode Standard, version 12.1)

Range: 1780-17FF

	178	179	17A	17B	17C	17D	17E	17F
0	ក 1780	ថ 1790	ហ 17A0	◌ 17B0	◌ 17C0	◌ 17D0	០ 17E0	๐ 17F0
1	ខ 1781	ទ 1791	ឡ 17A1	◌ 17B1	◌ 17C1	◌ 17D1	១ 17E1	๑ 17F1
2	គ 1782	ធ 1792	អ 17A2	◌ 17B2	◌ 17C2	◌ 17D2	២ 17E2	๒ 17F2
3	ឃ 1783	ន 1793	អ 17A3	◌ 17B3	◌ 17C3	◌ 17D3	៣ 17E3	๓ 17F3
4	ង 1784	ប 1794	អា 17A4	KIV AQ 17B4	◌ 17C4	◌ 17D4	៤ 17E4	๔ 17F4
5	ច 1785	ផ 1795	ឥ 17A5	KIV AA 17B5	◌ 17C5	◌ 17D5	៥ 17E5	๕ 17F5
6	ឆ 1786	ព 1796	ឦ 17A6	◌ 17B6	◌ 17C6	◌ 17D6	៦ 17E6	๖ 17F6
7	ជ 1787	ភ 1797	ឧ 17A7	◌ 17B7	◌ 17C7	◌ 17D7	៧ 17E7	๗ 17F7

[11] https://unicode.org/charts#Khmer


8 ឈ 1788 ម 1798 ឨ 17A8 ◌ុ 17B8 ◌ៈ 17C8 ฯលฯ 17D8 ៨ 17E8 ᧠ 17F8
9 ញ 1789 ឃ 1799 ឩ 17A9 ◌ូ 17B9 ◌៉ 17C9 ๐ 17D9 ៩ 17E9 ᧡ 17F9
A ដ 178A រ 179A ឪ 17AA ◌ួ 17BA ◌៊ 17CA ๛ 17DA
B ឋ 178B ល 179B ឫ 17AB ◌ើ 17BB ◌់ 17CB ៓ 17DB
C ឌ 178C វ 179C ឬ 17AC ◌ឿ 17BC ◌៌ 17CC ๘ 17DC
D ឍ 178D ឝ 179D ឭ 17AD ◌ៀ 17BD ◌៍ 17CD ๙ 17DD
E ណ 178E ឞ 179E ឮ 17AE ◌េ 17BE ◌៎ 17CE
F ត 178F ស 179F ឯ 17AF ◌ែ 17BF ◌៏ 17CF