Improving Sinhala OCR using Deep Learning

K.L.N.D.Liyanage

Improving Sinhala OCR using Deep Learning

K.L.N.D.Liyanage

Index No : 14000776

Supervisor: Dr. A.R. Weerasinghe

December 2018

Submitted in partial fulfillment of the requirements of the

B.Sc. (Hons) in Computer Science Final Year Project (SCS 4124)

Declaration

I certify that this dissertation does not incorporate, without acknowledgement, any material previously submitted for a degree or diploma in any university and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, be made available for photocopying and for interlibrary loans, and for the title and abstract to be made available to outside organizations.

Candidate Name: K.L.N.D. Liyanage

....

Signature of Candidate   Date:

This is to certify that this dissertation is based on the work of Ms. K.L.N.D. Liyanage under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Principle Supervisor's Name: Dr. A.R. Weerasinghe

....

Signature of Supervisor   Date:

Co- Supervisor's Name: Mr. K.V.D.J.P. Kumarasinghe

....

Signature of Co-Supervisor   Date:

Abstract

Converting a printed document into a stream of characters using optical character recognition techniques is a widely researched area. However, the unique cursive property of Sinhala characters makes character recognition a challenge. An improvement in Sinhala OCR is a much-needed requirement since this could benefit applications such as digitizing printed documents and books, data entry for business documents and, assistive technology for blind and visually impaired users among others. Even though accuracies of over 80% has been reported in previous studies, these have considered only a subset of Sinhala characters. In, this study we consider all the Sinhala characters including complex characters.

Since languages such as English and other, Latin-based languages have achieved state-of-the-art accuracies in character recognition using deep learning, their application to improve the Sinhala optical character recognition of printed characters has not been explored. A contour-based segmentation method is used in this research to segment Sinhala characters and its output recognized using a Convolutional Neural Network. An overall accuracy of 85.37% was achieved for segmenting and recognizing Sinhala characters. In summary, convolutional neural network-based model is capable of improving the Sinhala printed character recognition.

Preface

This dissertation proposes a deep learning approach to recognize Sinhala printed characters. According to the publicly available studies, a deep learning approach to recognize printed Sinhala characters has not proposed by any authors previously except for the recognition of ancient Sinhala inscription letters.

Out of the character segmentation techniques that was experimented in this study, horizontal and vertical projection profile-based character segmentation technique has been used in previous studies for recognize Sinhala characters. The concept behind the other two-character segmentation techniques has used in other studies for character recognition purposes as well as for segmentation purpose. The deep learning model was trained using the data set created by myself. Previous studies have considered only the recognition of subset of Sinhala characters. However, this study introduced a deep learning approach to recognize all Sinhala characters including complex Sinhala characters.

In order to propose a suitable deep learning approach, convolutional neural network architectures that is used in this study for experiments are extract from well-known image recognition and object detection CNN architectures suggested by previous studies. However, final proposing convolutional neural network architecture proposed in this study was solely my own work.

Acknowledgement

I would like to express my sincere gratitude to my supervisor, Dr. A.R.Weerasinghe, senior lecturer of University of Colombo School of Computing and co-supervisor Mr. K.V.D.J.P.Kumarasinghe for providing me constant guidance and supervision throughout this research.

I also appreciate the feedback and motivation given by my friends to achieve my research goals. I would also like to thank Mr. Chamila Liyanage, research assistant at Language Technology Research Laboratory in University of Colombo School of Computing for giving a hand on the research. Finally, my special gratitude goes to my loving family who has been an immense support to me throughout this journey of life.

Table of contents

Declaration **i**

Abstract **ii**

Preface **iii**

Acknowledgement **iv**

Table of contents **v**

List of figures **viii**

List of tables **ix**

Listings **x**

List of Acronyms **xi**

1 Introduction **1**

1.1 Background to the research 2
1.2 Research problem and research questions 3
1.3 Justification for the research 3
1.4 Methodology 4
1.5 Outline of the dissertation 5
1.6 Definitions 5
1.7 Delimitations of scope 6
1.8 Conclusion 6

2 Literature Review **7**

2.1 Introduction 7
2.2 Image preprocessing 7
2.3 Recognition process 9

2.3.1 Statistical method 9
2.3.2 HMM model 9
2.3.3 Fuzzy logic based method 10
2.3.4 Neural network based method 10
2.3.5 Deep learning based methods 11
2.3.6 Summary 13

3 Design 15

3.1 Introduction 15
3.2 Research design 15
3.2.1 Dataset 16
3.2.2 Image preprocessing 18
3.2.3 Character segmentation 19
3.2.4 Character recognition 22

4 Implementation 24

4.1 Introduction 24
4.2 Tools and technologies 24
4.3 Character segmentation implementation 25
4.3.1 Character segmentation by identifying contours 25
4.3.2 Character segmentation using convex hull 27
4.3.3 Horizontal and vertical projection profile-based segmentation 28
4.3.4 Character recognition models implementation 29

5 Results and evaluation 35

5.1 Introduction 35
5.2 Character segmentation 35
5.2.1 Character segmentation by using contours 35
5.2.2 Horizontal and vertical projection-based character segmentation 36
5.3 Character recognition 38
5.3.1 Experimental setup 38
5.3.2 Character recognition performance 39
5.3.3 Summary 44

6 Conclusions **45**
6.1 Introduction 45
6.2 Conclusions about research questions 45
6.3 Conclusions about research problem 46
6.4 Limitations 47
6.5 Implications for Further Research 47

References **48**

A Results **51**

List of figures

1.1 Proposed research methodology 5

3.1 Highlevel architecture of the system 16
3.2 Vertical segmentation of characters 17
3.3 Image preprocessing overview 18
3.4 Horizontal projection of the script 20
3.5 Segmented lines and its vertical projection 20
3.6 Output of contour-based character segmentation 21
3.7 Output of convex hull-based character segmentation 22

5.1 Sensitivity value of character recognition according to the font size 43
5.2 Output after mapping predicted classes for the image 4 44

A.1 Test Image 1 (script with font size 12 “Iskoola Pota” font) 51
A.2 Test Image 2 (script with font size 12 “Iskoola Pota” font) 52
A.3 Test Image 3 (script with “Nirmala” font) 53
A.4 Test Image 4 (Scanned image) (a) Original Image (b) Output after mapping 54
A.5 Test Image 5 (Scanned image) (a) Original Image (b) Output after mapping 55
A.6 Confusion matrix of the proposed classifier 56
A.7 Precision, recall, f1-score for approach 1 57
A.8 Precision, recall, f1-score for approach 1 58
A.9 Precision, recall, f1-score for approach 1 59
A.10 Precision, recall, f1-score for approach 1 60

List of tables

5.1 Accuracies for character segmentation using contour based segmentation . . . 36
5.2 Accuracies for horizontal and vertical projection-based character segmentation . . . 37
5.3 Results of font size combinations for each architecture . . . 38
5.4 Cross validation accuracy for each CNN architecture . . . 39
5.5 Accuracy of the automatically and manually segmented characters . 40
5.6 Overall accuracy of the classifier including segmentation accuracy . 42

Listings

4.1 Preprocessing steps used in contour-based segmentation 26
4.2 Finding contours in script 26
4.3 Sort the contours to get the order of characters 27
4.4 Finding convex hull of characters 28
4.5 Obtaining the lines in the script 28
4.6 Character segmentation from words 29
4.7 AlexNet architecture 30
4.8 LeNet architecture 32
4.9 ResNet architecture 32

List of Acronyms

CNN Convolutional Neural Network.

DBN Deep Belief Network.

DPI Dots per Inch.

LSTM Long Short Term Memory.

OCR Optical Character Recognition.

Chapter 1

Introduction

Printed media plays a major role when it comes to sharing of documents all over the world. Digitizing these printed documents eliminates the need for retyping already printed documents for editing. Optical Character recognition (OCR) is the technology that is used to convert information available in the printed form into machine editable electronic text through a process of image capture, processing and recognition [1].

Sinhala Scripts has been descended from the ancient Brahmi script and evolved independently over many centuries. The Sinhala language is unique to Sri Lanka and the characters are differ from all the other descended scripts in South Asia due to the unique cursive property in characters. Sinhala alphabet consist with vowels and modifiers. A vowel may appear only as the first character of a word and a consonant is modified using one or more of the modifier symbols to produce the required vocal sound. The total number of different modifications from the entire alphabet including the basic characters is nearly 400 [2].

Cursive property of Sinhala characters makes character recognition process challenging task. However, most of the existing OCR systems for languages such as English, Latin-based languages have achieved state-of-the-art character recognition accuracies using Deep Learning [3]. This technology is being used in various fields including Natural Language Processing and has been showed significant improvement in solving problems. A prime example for this is, deep learning models used to recognize characters.

In this study we exploit the use of deep learning for recognizing Sinhala printed characters.

1.1 Background to the research

Optical character recognition for Sinhala printed characters as well as for handwritten characters is still an evolving research field. As mentioned in previous section, due to the unique cursive property of Sinhala characters, character recognition process is a challenging task. In spite of the title being the Optical Character Recognition, the work of digital image processing also has to be done in this study.

According to the previous studies, OCR process consists of three main stages: Preprocessing, Recognition process and Postprocessing [1]. Preprocessing stage has been used to improve the quality of the scanned image through various image processing techniques such as image binarization, noise removing, skew detection and correction, normalization and segmentation [3],[4], [5]. Image processing techniques that are applied in this stage directly influence the character segmentation step [3]. Therefore, one of the significant step in this study is image preprocessing. Character segmentation has played another important role in character recognition since features of segmented character is the input to the recognition model.

Second important stage in OCR is, character recognition process. Features of the characters are extracted from the preprocessed image and input into the character recognition system. In the field of OCR for Sinhala language, various approaches are being used for recognition process. Rule based methods, Stochastic methods [4], Machine learning and neural network methods [3] and Hybrid methods [6] are some of the approaches taken by researchers. However, most of the studies that have achieved higher accuracy rates have considered only a subset of Sinhala characters [4], [5]. Even though a number of studies have been done to recognize both printed and handwritten Sinhala characters, no record has been identified for recognizing Sinhala printed and handwritten characters using deep learning approaches. The only record of using a deep learning approach was, recognizing ancient Sinhala inscription letters [7].

Post processing stage has been used to improve the results and accuracy of OCR by using various methods like using a look up table to map characters, maintain a dictionary including all the words in particular language, keep a text file to map the output with the corresponding character [3] etc.

1.2 Research problem and research questions

Most of the existing OCR systems for English language and Latin-based languages have achieved state-of-the-art character recognition accuracy rates. However, among other languages, characters of Sinhala language are unique mainly because they are round in shape. This unique feature makes it a challenge to extend some existing techniques to improve recognition of Sinhala characters [3].

Approaches that have been proposed by previous studies, have failed at recognizing similar characters [4], English characters in documents [8] and some font sizes. As Sinhala OCR performance is not good on unrestricted text such as similar characters, different font types and font size, English characters, this research will focus on improving the performance of Sinhala OCR while improving the recognition of unrestricted text.

Since OCR on other languages have been benefited from deep learning, this research is an attempt to design a deep learning network to recognize Sinhala text more accurately [9], [10].

Considering this problem, the generated research questions are as follows.

- Can deep learning be used to improve the current performance of Sinhala OCR?

  o What is the effect of character segmentation technique for recognizing characters?

  o What is the effect of the font size of the training character data set for recognizing characters?

1.3 Justification for the research

Optical character recognition systems have failed in recognizing unrestricted text such as similar characters like 'ත', 'න', 'ක' [4], some font types etc. Therefore, OCR accuracy is not in a higher accuracy range with compare to the OCR systems for languages like English, Latin-based languages. Even though some of the existing studies for Sinhala printed character recognition have achieved accuracy rates within 90% - 100% range [5], [11], these studies have been only limit for a subset of Sinhala character recognition, mainly for the Sinhala core characters. Hence, there is a necessity of improving the Sinhala optical character recognition as having more accurate OCR system benefit application domain such as digitizing printed documents and books [3], data entry for business documents, as an assistive technology for blind and visually impaired users [1] etc.

Many Sinhala character recognition approaches have been used rule-based approaches, stochastic approaches [4], machine learning and neural network methods [3], hybrid methods [6]. They haven't been used a deep learning approach for printed Sinhala character recognition. Recently, one of the studies have used a deep learning approach to recognize ancient Sinhala inscription characters [7]. However, in this study they have only considered basic Sinhala characters and the data set mostly include synthetic data created using a photo editing tool. Thus, this study focused on applying a deep learning approach to recognize printed Sinhala characters.

As mentioned earlier, most of the researchers focused on subset of Sinhala characters. Thus, this research focuses on achieving a performance improvement of Sinhala printed character recognition while including all the complex Sinhala characters with character modifiers.

1.4 Methodology

The first step of this study is to analyze the existing approach to recognize Sinhala characters and identified the key finding of this approaches that can be taken. Creating training data set of character images is considered as the next step of this proposed approach. Then appropriate image processing techniques are identified to enhance the image quality. The next step is to review character segmentation techniques that can be applied. After exploring character segmentation techniques, deep learning approaches used in other proposed character recognition solutions are studied and devise a deep learning approach to recognize Sinhala printed characters. The final step is focused on evaluating the performance of the proposed approach in this study.

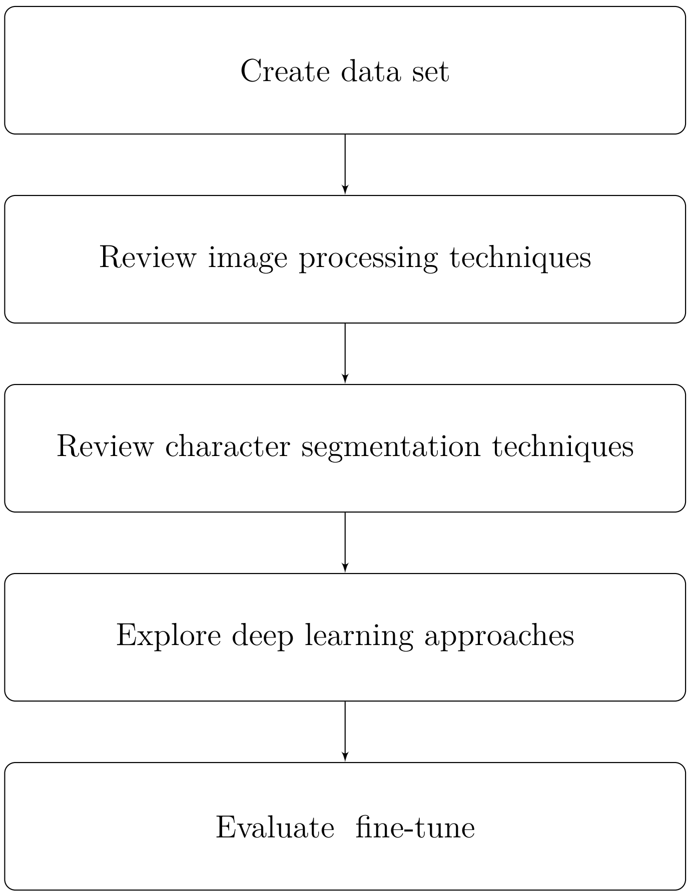

Create data set
Review image processing techniques
Review character segmentation techniques
Explore deep learning approaches
Evaluate fine-tune

Figure 1.1: Proposed research methodology

1.5 Outline of the dissertation

The dissertation is structured as follows. Chapter two presents the existing approaches related to the domain of Sinhala printed and handwritten character recognition. Further, this chapter describes the deep learning approaches used in character recognition. Chapter three describes the proposed research design. Implementation details of the proposed methodology is discussed in the chapter four. Chapter five presents the evaluation results of the proposed approach. The last chapter, chapter six demonstrates the conclusion of the thesis and outline of the future work.

1.6 Definitions

In this study, accuracy given by the proposed deep learning approach is measured by character recognition accuracy. Character recognition accuracy can be defined as percentage of characters that are correctly recognized in a given script. Furthermore, segmentation accuracy is measured using ten scripts with different font types. Percentage of characters correctly segmented from the total number of characters in the script, is given as the segmentation accuracy of each script.

1.7 Delimitations of scope

A deep learning approach to recognize Sinhala printed characters will be introduced in this study. This research is focused on recognizing commonly used Sinhala fonts. Sinhala ancient non-digital fonts are not considered in this study. Recognizing the fonts with font size 12 and above will be considered since some fonts and Sinhala characters like ඏ below 12px is hard to recognize even for the human. Scanned images of printed documents with 300 dots per inch value or above will be considered when proposing the approach.

1.8 Conclusion

This chapter introduced the research problem and research questions and hypotheses. Then the research was justified, definitions were presented. The methodology was briefly described and justified in the subsequent sections. The dissertation was outlined, and the limitations were given in the section 1.5 and 1.7. On these foundations, the dissertation can proceed with a detailed description of the research.

Chapter 2

Literature Review

2.1 Introduction

In this chapter, review of related work on optical character recognition of Sinhala printed characters as well as the handwritten characters is provided. Recent attempts to recognize Sinhala characters in printed scripts as well as the handwritten characters have been depicted that character recognition depend on factors such as approach used to identify characters, quality of the scanned image that has been used as the input to the model. Thus, Section 2.2 focuses on image preprocessing techniques that have been carried out to enhance the image quality and Section 2.3 presents a review of the existing approaches for character recognition process.

2.2 Image preprocessing

The image should be undergone a preprocessing stage in order to reduce unwanted pixels and to differentiate foreground and background in a way to support character segmentation process. Many techniques have been used to enhance the quality of the image in order to improve the recognition of characters. Image binarization, noise removing, skew detection and correction, normalization and segmentation are the common preprocessing techniques followed by previous studies [3], [4], [6]. Noise removing can be done using smoothing techniques and as well as thresholding techniques. There are two main types of noise presets are in images; Gaussian noise and salt/pepper noise. Gaussian noise can be reduced using Gaussian filtering algorithm, while median or mean filtering algorithm could be used to filter salt and pepper noise. In [12] researchers have been used median filtering algorithm with a 5X5 kernel to remove the noise. Image binarization using global thresholding technique have been used in [4] to eliminate noise. In thresholding, low intensity pixels will be removed, while only retaining the pixels which belong to the actual characters. In this step, all the pixels below a specific threshold value will be set to white color. In [3] and [6] also, other image preprocessing techniques has followed image binarization using thresholding techniques.

Other than image binarization and smoothing, image preprocessing techniques like dilation, opening and closing have been followed in some studies. In research [3], connected components detection and image dilation has been used to the enhance the quality of the images after binarization. In this research, they have cropped the glyphs into individual components before applying dilation in order to prevent character merging in dilation step. In dilation, pixels are added to the boundaries of objects in an image depends on the shape and size of the structuring element used to process the image. Though the researchers have followed this step, due to the rounded shape in characters still there were few merging characters that has caused difficulties in character recognition. In [6], image binarization step has followed by other image preprocessing techniques like opening to remove noise, closing to fill the holes in the letters, skeletonized to obtain the real shape of the letter, spur reducing, resizing letter into 7:5 ratio and image dilation. Opening is dilation of erosion. In simple terms, image first gets eroded and then gets dilated depending on the structuring element used in the process. In dilation, pixels are removed from boundaries of the objects in an image. In closing, dilation is followed by erosion [13].

Character segmentation is the next process that take place prior to character recognition. Accuracy of this step, directly affects the accuracy of character classification step. To segment characters various techniques have been followed by previous studies. One of the approaches commonly used is character segmentation using horizontal and vertical projection profiles. In this approach vertical profile is calculated by summing the black pixels perpendicular to the y axis and horizontal profiles is calculated by summing the black pixels perpendicular to the x axis. In [4], they have been used horizontal projection to segment words and characters. Lines of the scripts have been segmented using vertical projection, before applying horizontal projection. Same approach has been taken by Ajward *et al.* [3] to segment script into lines, words and characters. In [6], researchers haven't mentioned segmentation techniques used to segment scripts into lines, words and characters. In the paper [14], C. Silva *et al.* has been proposed a novel approach to segment characters using Self Organizing Feature Maps as a solution to the problem touching character pairs. Premaratne *et al.* have proposed a novel approach to segment characters by using the orientation features of characters directly using a standard alphabet as the basis without the need for segmentation into basic components [15]. Though this approach does not have segmentation complexity, many iterative filtering should be performed.

2.3 Recognition process

Many approaches have been taken by researchers to recognize characters and its features. Stochastic approach [4], [11], fuzzy logic approach [6], machine learning and neural network approach [3], hybrid approach and deep learning approach [7] have been taken by various studies.

2.3.1 Statistical method

[4] has proposed a statistical method to recognize Sinhala handwritten characters. Statistical classifier based on interval estimation has been used in [4] to distinctly identified Sinhala handwritten characters. Hewavitharana *et al.* have mentioned that in handwriting recognition point of view, characters can be visualized in terms of three vertical zones. Depending on the position of the character with respect to the three-zone frame, each character has been pre-classified into six groups prior classifying by statistical classifier. Each character has been resized into 32 x 32 pixels common height and width using bilinear interpolation technique and 71-dimensional feature vector was being extracted from these resized images. Hewavitharana *et al.* have further stated that pre-classification method used in this study would have much higher recognition accuracy if applied to recognition of printed scripts. However, this proposed method has been failed when recognizing similar characters like 'ට', 'ව', 'ම' and 'ත', 'න', 'ක'. .

2.3.2 HMM model

Similar approach to [4] has been taken in [11]. In this study they have been used a discrete hidden Markov model (HMM) based classifier to recognize characters. Like [4], they have used a pre-classifier to classify characters prior applying statistical classifier. Unlike [4], which has used six group classification, this method has been used the pre-classifier to classify characters into three groups; Core characters, Ascending characters and Descending characters. Bilinear interpolation has been used to resized characters into common height and width like in [4]. Then the image has been divided into horizontal and vertical strip and each strip is subdivided into sections of size 4x4 pixels. Pixel density of each section has been used to create a feature vector in two directions: horizontally and vertically. Character recognition has been done by adding log probabilities for each character calculated in horizontal and vertical direction using HMM. According to the researchers, feature set selected by their method has been under represented the character classes and hence it has been led to low character recognition. This study, has been depicted that chosen feature set has direct influence to the recognition of characters.

2.3.3 Fuzzy logic based method

Fuzzy logic-based approach has been taken in [6] to recognize characters which make use of unique cursive property of characters. First feature extracted from the character was the distance from the center to the edges of the letter along eight directions. Using these distance measurements, rules and input/output membership functions has been defined and using them Fuzzy Inference System mapped the fuzzy inputs with the fuzzy outputs. The second feature extracted from the characters were the number of intersections from the center of the letter to the eight directions. These features have been used as the input to a second Fuzzy inference system and the system has mapped the fuzzy inputs with the fuzzy outputs using the rules and membership functions defined according to the intersection measurements. Characters were recognized upon the optimal crisp value returned from either Fuzzy inference system 1 and 2. This system has been tested using three datasets to measure the accuracy of basic character recognition, similar characters recognition and elapsed time. Unlike Statistical based method [4] and HMM model-based method [11], this method has been achieved reasonable accuracy in recognizing similar characters.

2.3.4 Neural network based method

[3] has proposed a character recognition method using feed forward back propagation neural network with two hidden layers. From this study they have developed a method of preserving a number of selected formatting features of a printed document. To recognize font attributes, projection profile of text lines has been used. Font size has been discriminated by measuring the height of the vertical profile. Normal, Bold, Italic, and Bold Italic fonts has been identified using the variance of the derivative of the horizontal profile. Continuity of vertical profile has been used to discriminate underline text. In other studies, formatting features of printed document has not been considered when developing a method to recognize characters.

2.3.5 Deep learning based methods

Even though a number of studies have been done to recognize both printed and handwritten Sinhala characters, no record has been identified for recognizing Sinhala characters using deep learning approaches. However, a deep learning approach has been taken in [7] to recognize ancient Sinhala inscription characters. In this section, studies that have been conducted character recognition using deep learning models has explained for ancient Sinhala inscription letters as well as for other Languages.

2.3.5.1 Convolutional neural network based methods

In this study [7], they have been used neural network approach and the convolutional neural network approach to recognize characters and evaluate the recognition accuracy for each approach. According to their results, CNN based OCR has shown better accuracy than neural network approach. They have been conducted an experiment to measure the accuracy of noisy images recognition and, from their experiment they have stated that when the noise rate of the input images was getting higher, recognition rate was decreased. Further they have stated results of the characters recognition has been depend on the quality of the input image. From the results of this study, it can be seen that deep learning approach for OCR is a path that worth being pursued. Even though a deep learning approach has not been taken to recognize Sinhala printed scripts and handwritten characters, there have been several attempts taken in descendant languages of Brahmi language. Details of these attempts have been explained in subsequent paragraphs.

In research [16], character recognition using LeNet-5, a convolutional Neural Network trained with gradient based training and backpropagation algorithm was used for classification of Malayalam character images. Results obtained from their study have been showed that, CNN performance had dropped down when the number of classes exceeds range of 40. Therefore, they have followed a multi stage classification. CNN has been used to primarily for the recognition and misclassified characters has been further classified using multiclass SVM.

Research [17] has been taken approach to recognize handwritten words using two approaches; classifying words directly and character segmentation. In this approach they have used CNN for character recognition purpose and LSTM (Long Short-Term Memory) network to character segmentation. Character segmentation has been done by using Tesseract 4.0 version which is neural network-based CNN/L-STM network. First, they have experimented word level classification and it has been given low accuracy rates. Hence, this study has classified each character independently to reconstruct whole word. From those two experiments they have been concluded that character level training improves performance than word level training. Further, researchers have suggested to use large mini batch size since it would capture too much information at once and fasting the model. This research has been depicted the effect of character segmentation to the performance of the recognizer.

2.3.5.2 Deep belief network based models

[9] has used unsupervised stacked Restricted Boltzmann Machines (RBM) to learn features and a Deep Belief Network to recognize handwritten Devanagari characters. In this study Prabhanjan *et al.* has done several experiments to decide on the number hidden layers to be included in the network. From the result they have concluded that having a few hidden layers have been led to short training time period but resulted in poor performance as the system cannot be fully store all the features of the training data sets. Further they have stated that too many layers may result in over-fitting and slow down the learning time. Therefore, they have done experiments for three different settings for number of hidden layers by keeping other parameters as constants. Performance of proposed method for unsupervised learning, has been evaluated for numerals, vowels, consonants, compound characters and vowel modifiers individually and combining both characters and numerals. From the results, researchers have concluded that since the experiments were done using unsupervised learning, accuracy was lesser. Therefore, they have fine-tuned the model using supervised learning.

In [10], Bangla handwritten character recognition has been presented by excluding the burden of feature extraction. This study has used a deep belief network (DBN) to recognize characters with unsupervised learning followed by supervised fine tuning of the network. Raw image data is the input to the DBN. Unsupervised learning was performed by using contrastive divergence algorithm which is an approximation of maximum likelihood estimation, while in the later stage network parameters are fine-tuned with the gradient based backpropagation algorithm. Contrastive divergence algorithm was used since maximizing the likelihood requires a Markov chain Monte Carlo simulation which takes a long time. One of the objectives of Biswas *et al.* was the evaluation of the effectiveness of unsupervised feature learning approach using DBN. Hence, in order to see the effect, they have been trained another DBN without unsupervised feature learning but only conjugate gradient back propagation has been used. Results of the experiment has been showed that unsupervised learning had led to significant improvement recognition. One of the prime features of the DBN approach is the ability of image reconstruction as it can work as a generative model. Researchers have compared their results with hierarchical learning architecture, who reported their performance on the same data set that used in their experiment. Experiment has been showed that DBN can achieve higher recognition rates even though no handcrafted features was used. This study has stated that if they could use larger data set than used data set, results would be better than this.

2.3.5.3 Recurrent neural network with LSTM based models

Shkarupa *et al.* [18] have been used a Recurrent Neural Network with Long Short-Term Memory for the recognition of historic handwritten Latin texts. This study has used two RNN architectures; Connectionist Temporal Classification and Sequence to Sequence Learning approach. Unlike [17], this study has been focused on the simplicity of the architecture in order to complete the classification within reasonable time using an average PC. In this study optimal configurations for each performance has been chosen by conducting several experiments. Mini batch approach was used as training approach and this study has stated that use of weight cross-entropy loss function reduced the convergence time of the model. They have been experimented the use of a CNN for feature extracting and the results have been showed no improvement in recognition accuracy. In this study, performance has been evaluated using two metrices: Word accuracy rates and Character accuracy rates.

2.3.6 Summary

This section has been explained various attempts taken by researchers to recognize both Sinhala printed scripts and handwritten characters. Statistical approach, neural network approach, HMM model-based approach and Fuzzy logic-based approach have been discussed in terms of image preprocessing techniques followed and the recognition process used. When analyzing those studies, several drawbacks and research gaps can be recognized such as difficulties in identifying similar characters, not able to recognize unrestricted text as mentioned in the section 1.3 and only a subset of Sinhala characters has considered in the studies. Even though a number of approaches have been taken, no record for recognizing Sinhala printed scripts and handwritten characters. Hence, literature review has done to analyze the studies that used deep learning approach in OCR of several other languages. However, recently a research [7] has been conducted using deep learning approach to recognize ancient Sinhala inscription characters. This study has been shown the performance improvement in character recognition using deep learning approach over artificial neural network. Studies such as [19] has been shown the performance improvement in OCR when feature learning is done using deep learning approach rather than using handcrafted features for recognition. Thus, all the deep learning studies for OCR discussed in this section has been depicted that deep learning approach for OCR is a path that worth being pursued.

Chapter 3

Design

3.1 Introduction

This chapter explains the proposed solution for the research problem. Section 3.2 explains the research design while the subsequent sections explain the main step performed in the research design.

3.2 Research design

The system consists of separate individual components based on the different functions needed to be performed in the process of an OCR. These include preprocessing, character segmentation, character classification and post processing. The high-level view of the architecture of the whole system and the connectivity of the separate components has been shown in following figure 3.1 Main source of input to the system would be an image of a printed script and the output from the system would be recognized character.

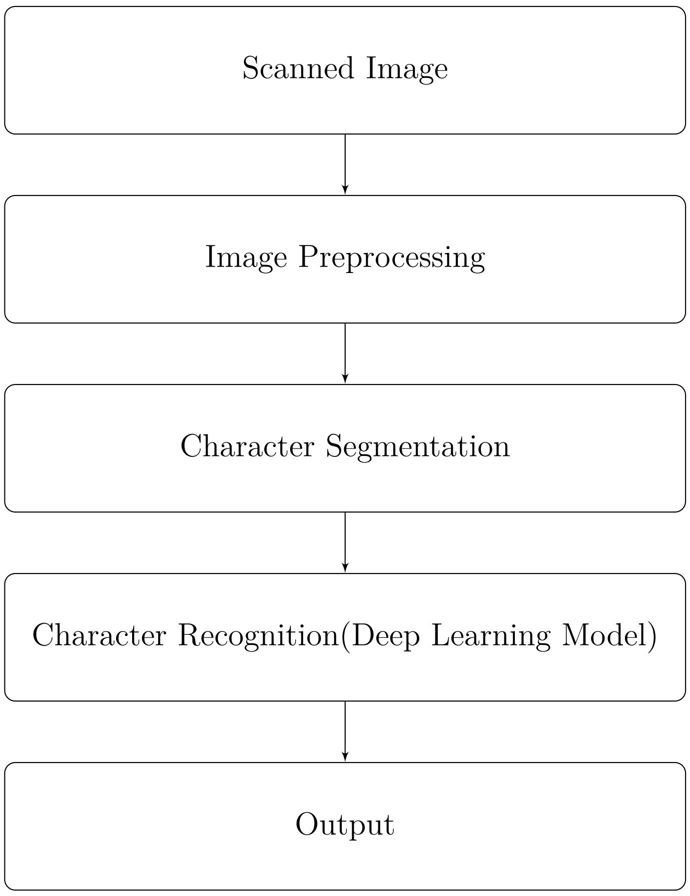


Figure 3.1: Highlevel architecture of the system

Above figure represents the steps involved in the construction of the system and the flow of data. Initially, scanned image with 300 dots per inch value (DPI) or above, will be preprocessed to enhance the quality of the image using image preprocessing techniques. The characters in the preprocessed image will be segmented as the next step and the raw image pixel value of segmented characters are used as the input to the deep learning model. Feature extraction and selection processes are performed by the deep learning model. Recognized character will be given as the output from the model. The system is trained using Sinhala printed characters before it is used to recognize characters. Subsequent sections will be explained the data set creation process, image preprocessing process, character segmentation process and the character recognition process.

3.2.1 Dataset

Training data set creation was the first step that was carried out in this study, before considering about the preprocessing stage. The model that is proposed by this study, was trained using printed Sinhala characters. Character level training is performed since training with each and every Sinhala word couldn't be practically done as Sinhala language has a rich vocabulary and the results of the study [17] shows that character level training increased the character recognition accuracy than the word level training. Creating a character image set cannot be practically done in this study since using scanned images since the parameters such as font size and font type which is needed to address the second research question in this study cannot be measured. Hence, this study has focused on creating training character images by converting screen text into jpeg format images. The characters are typed in a word processing software by using three font sizes; 12, 14, and 16px and eight font types. Eight commonly using font types were used for the training in order to increase the size of the training data set since in the deep learning models, larger the data set accuracy get better. Then the screenshots were converted into jpeg format images. Further, distorted images of characters were also added to the dataset as in practical scenario different variance can be happened due to the brightness, lighting condition, orientation etc.

After creating images of scripts, each character has been segmented from the script by preserving the aspect ratio. Character images were padded by white pixels in order to make all the character images into common size. Therefore, to segment characters from the script without manual intervention, characters segmentation techniques were implemented and used in this stage. The images of scripts were vertically segmented into individual characters as shown in Figure 3.2. Only the vertical segmentation of characters was performed since horizontal segmentation of characters is a difficult task as horizontal modifiers of Sinhala characters are connected with the base characters. Character classes labeling was carried out in this step, since supervised learning approach will be taken in this study. Since vertical character segmentation was carried out, there were altogether 297 - character classes including the main punctuation marks; full stop, comma, and brackets in the created training data set.

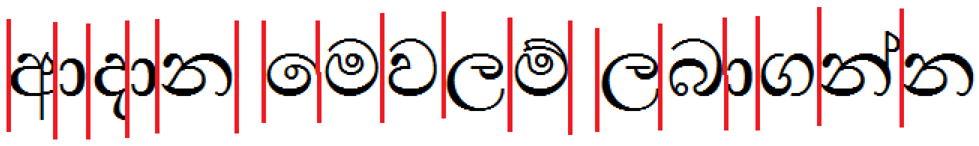


Figure 3.2: Vertical segmentation of characters

3.2.2 Image preprocessing

This stage is one of the important stages in every OCR process, since preprocessing should be done in a way to enhance the performance of character segmentation. Several image preprocessing techniques has been applied in order to enhance the quality of the image such as smoothing, thresholding, correcting image orientation and scaling. Figure 3.3 shows the image preprocessing techniques used in this study to enhance the image quality.

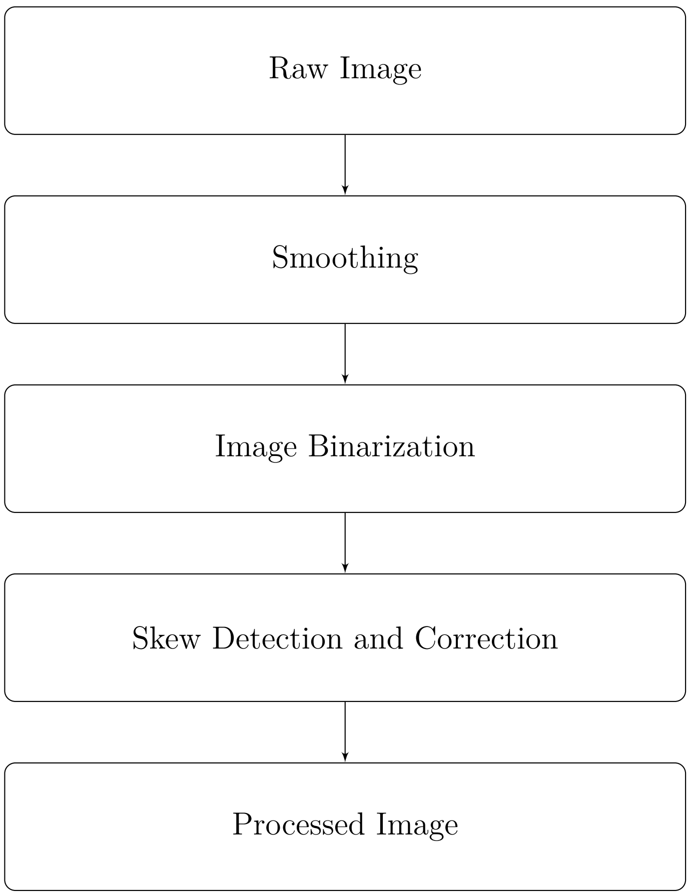


Figure 3.3: Image preprocessing overview

Image smoothing is applied for the raw images input into the system to remove noise in the image. In this stage, gaussian filter is used to remove the gaussian noise in an image. Noise removed image is binarized using image binarization technique, adaptive thresholding to reduce the complexity and make easy to do the segmentation. When the scanning process is doing, images could be scanned with varying illumination. Thus, the reason to use adaptive thresholding technique is, this method gives better results for images with varying illumination. Finally, since the scanned images could be skewed while in the scanning process, skewed images are detected and modification are made to those skewed images.

3.2.3 Character segmentation

Character segmentation is a main factor that affects the character recognition in an optical character recognition system. Therefore, in order to improve the character recognition process three different character segmentation techniques are experimented in this study.

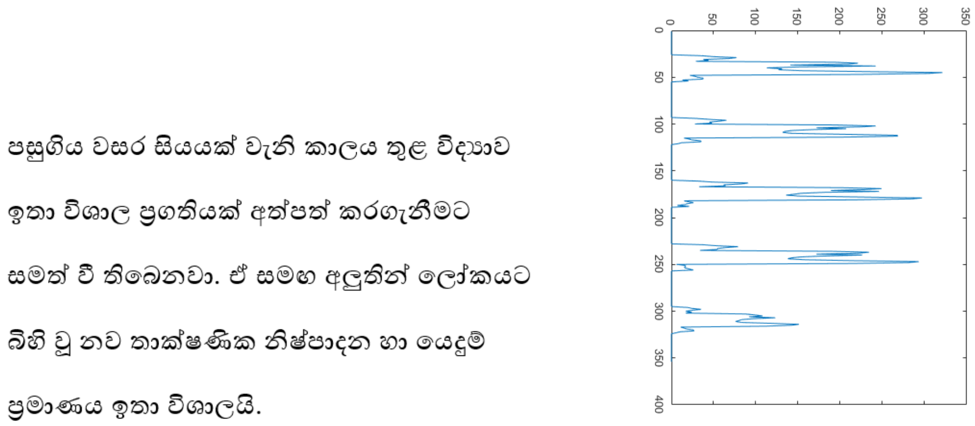
Character segmentation techniques are used to segment script into lines, words and characters after preprocessing images as well as to create the training dataset. Character segmentation techniques are experimented by using the techniques followed by related works in literature review as well as using the functions provided by the image preprocessing libraries. Techniques such as horizontal and vertical projection profile-based segmentation, contour-based segmentation and convex hull-based segmentation techniques are experimented and based on the effect for the performance improvement in overall system, better approach is selected to the final proposing solution.

Horizontal and vertical projection profile based segmentation
This technique is the mostly used character segmentation technique in previous attempts to recognize Sinhala characters. In this technique, horizontal projection of the scripts is used to segment script into lines. After identifying lines in the script, words and characters are identified respectively using vertical projection profile of the script. Projection of a binary image onto a line is obtained by partitioning the line into bins and finding the number of 1 pixels that are on lines perpendicular to each bin in horizontal and vertical direction. The horizontal projection H[i] along the rows and the vertical projection V[j] along the columns of a binary image are calculated by following equation 3.1 and equation 3.2 equations.

$$H[i] = \sum_{j=0}^{m-1} B[i,j] \qquad \text{(equation 3.1)}$$

$$V[J] = \sum_{i=0}^{n-1} B[i,j] \qquad \text{(equation 3.2)}$$

Figure 3.4 and Figure A.7 shows horizontal projection profile of the Sinhala script, vertical projection profile of the script and the segmented lines of the script respectively.



පසුගිය වසර සියයක් වැනි කාලය තුළ විද්‍යාව

ඉතා විශාල ප්‍රගතියක් අත්පත් කරගැනීමට

සමත් වී තිබෙනවා. ඒ සමඟ අලුතින් ලෝකයට

බිහි වූ නව තාක්ෂණික නිෂ්පාදන හා යෙදුම්

ප්‍රමාණය ඉතා විශාලයි.

Figure 3.4: Horizontal projection of the script

As illustrated in above figure, five peaks in the Figure 3.4 corresponding to the five lines in the script.

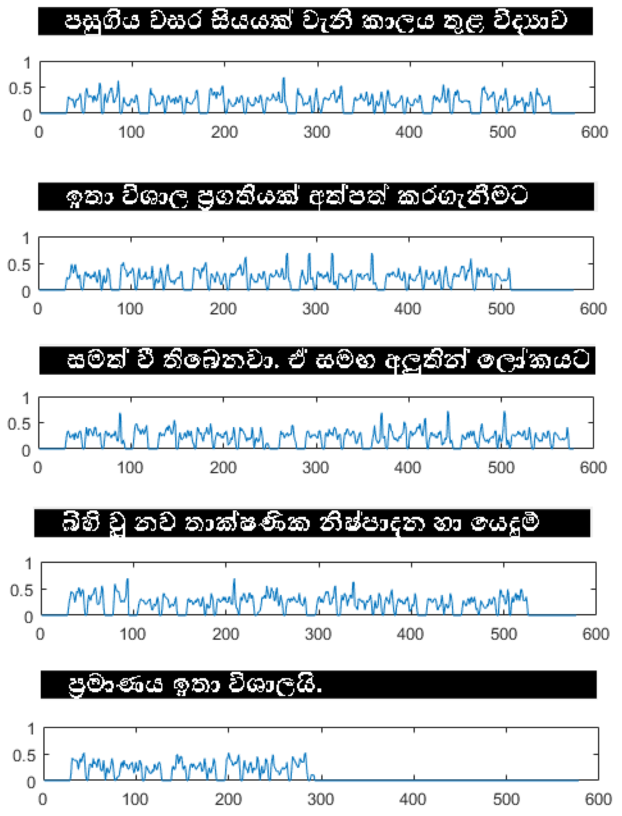
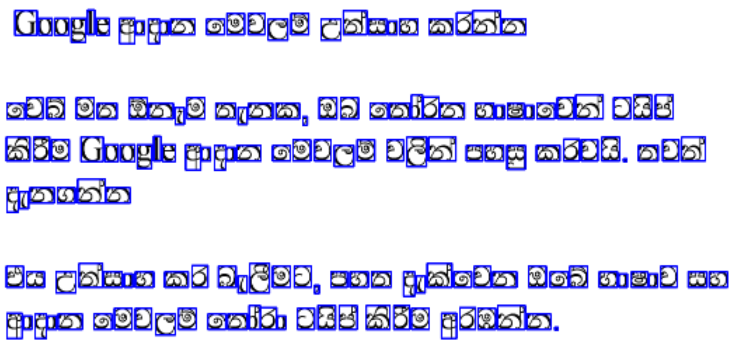


Figure 3.5: Segmented lines and its vertical projection

Character segmentation by identifying contours

This character segmentation technique has used contours along the boundary of characters to identify characters. Contour is a curve joining all the continuous points along the boundary having same color or intensity. After finding the contours in script, bounding box is drawn around the character. Character are then segmented by using x and y coordinates and the width and height of the bounding box. OpenCV 3.4.0 library is used in this segmentation technique. Implementation details have explained in the section 4.3. Following figure 3.6 shows output of a script that has used contour-based character segmentation.



Google ආදාන මෙවලම් උත්සාහ කරන්න

වෙබ් මත ඕනෑම තැනක, ඔබ තෝරන භාෂාවෙන් ටයිප් කිරීම Google ආදාන මෙවලම් වලින් පහසු කරවයි. තවත් දැනගන්න

එය උත්සාහ කර බැලීමට, පහත දැක්වෙන ඔබේ භාෂාව සහ ආදාන මෙවලම් තෝරා ටයිප් කිරීම අරඹන්න.

Figure 3.6: Output of contour-based character segmentation

Character segmentation using convex hull

This segmentation technique is used to draw a convex hull around the character by using OpenCV 3.4.0 library. Given a set of points in a plane, the convex hull of the set is the smallest convex polygon that contains all the points of it. Binarized image from the image preprocessing stage is taken as the input and first contours are find in the script. After finding the contours, then the convex hull is drawn around the character using the convexHull class in the OpenCV library. Finally, the character is segmented using the boundary given by convex hull.

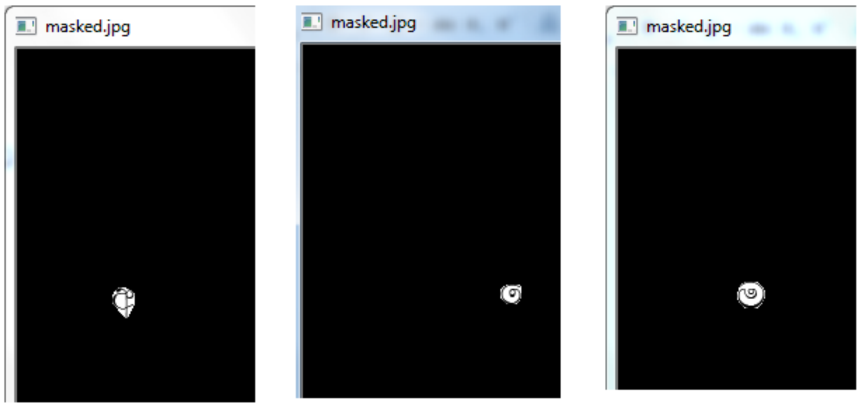


Figure 3.7: Output of convex hull-based character segmentation

3.2.4 Character recognition

Character recognition is the next step after segmenting characters. Convolutional Neural Network (ConvNet or CNN) is a special type of neural network used effectively for image recognition and classification. Hence, characters are recognised by using a convolutional neural network (CNN) since CNN shows impressive results in recognizing images with varying illumination and noise. Character recognition experiments are carried out by using the CNN architectures that won the image classification competition ImageNet Large Scale Visual Recognition Challenge. Therefore, several experiments are carried out using four architectures such as AlexNet [20], LeNet [21], GoogleNet [22] and ResNet [23] and the results of three of them were presented in this study.

Before input into the CNN model, input images were padded in order to create fixed size input images. Each character image is converted into fixed 28x28 pixel matrix and these raw pixel values are input into the model. In this study, raw image pixel values are used as the input since CNN tends to work better on raw input pixels rather than features or parts of an image [17]. Pixel values of each segmented character image is normalized since fixed size of input images will be used as the input. After normalization, pixel values of the image are used as the input to the model. In this study, since the feature engineering will be carried out by deep learning model, deciding on the feature set to be extracted is not a burden.

A suitable CNN architecture was proposed by obtaining the prediction accuracy using cross validation for each CNN architectures. Each CNN has been trained using the created dataset as explained in the section 3.2.1 and tested with images created like the training dataset.

Deep learning model will be given set of probabilities as output. Class of the higher probability value of given output will be considered as the class that character is belonged to. In order to map the predicted class label to the corresponding character, mapping has been made as the post processing step.

Chapter 4

Implementation

4.1 Introduction

This chapter elaborates the implementation details of the proposed solution. Section 4.2 describes the software tools utilized for the implementation and the section 4.3 contains the details about character segmentation implementations. Finally, section 4.4 contains the implementation details of character recognition models that are tested currently.

4.2 Tools and technologies

Scanned images from books and papers couldn't be used in creating character images as parameters such as font size and font type are impractical to measure in scanned images. Hence, screen text is converted into images, and Photoshop CS6 is used to set the DPI (dots per inch) value of images.

Python language is used to implement the contour based and convex hull-based character segmentation techniques and for other implementation tasks such as arranging the character images into fixed common height, generating raw pixel values of images. Main reason to use python is, image processing library like OpenCV which can be easily used to image binarization, skew detection and correction, morphological transformations are available in python.

Horizontal and vertical projection profile-based character segmentation technique was tested by using MATLAB. MATLAB is used in experiment since the Image Processing Toolbox in MATLAB provides a comprehensive set of reference-standard algorithms and work flow apps for image processing.

In deep learning, results get better with more data and larger models that in turn require more computation power. Therefore, GPU usage for these tasks is very useful as GPUs are efficient in processing large number of data in parallel. NVDIA provides a deep learning SDK for this purpose, which supports deep learning frameworks such as TensorFlow, Caffe, Theano etc. In this study, we used TensorFlow to build the model since it allows computations to be spread out across many computational devices across many machines and allows users to specify machine learning models using relatively high-level descriptions [24]. In order to build the deep learning model easily using TensorFlow, a high-level neural networks API Keras [25], written in Python was used.

Deep learning models are taking quite long time to train when use CPU power. Hence, in order to run deep learning model on GPU, Google Colab [26] is used as it supports free GPU.

4.3 Character segmentation implementation

This section contains the implementation details about three-character segmentation techniques.

4.3.1 Character segmentation by identifying contours

In this approach, as mentioned in the section 3.2.3, each character is identified by using contour tool in OpenCV library. The image is first binarized using thresholding technique, adaptive thresholding. As shown in listing 4.1 then the morphological operation dilation was applied for the binarized image in order to increase the thickness of the character. Then *findContours()* function in OpenCV library is used to identify contours in the script. In this function three parameters: source image, contour retrieval mode, contour approximation method is passed in order to get the contour array. The important parameter in this function is contour approximation method. Contour stores the (x,y) coordinates of the boundary of a shape. Contour approximation method limits the coordinates stored in contour array of each character. In this study, characters are vertically segmented as mentioned in the section 3.2.1. Therefore, in this scenario only the end points of characters are needed and by using *CHAIN-APPROX-SIMPLE* approximation method of OpenCV 3.4.0 and several other techniques, only the needed end points are taken. *CHAIN-APPROX-SIMPLE* method removes all redundant points and compresses the contour, thereby saving memory. After finding the contours, x, y, width and the height coordinates are obtained by using *boundingRect()* function. Since contour based method doesn't crop the characters in the order of the characters in the script, middle coordinate of the bounding box was used to identify the order of characters using *moments()* function as shown in listing 4.2.

Listing 4.1: Preprocessing steps used in contour-based segmentation

```python
#grayscale
gray=cv2.cvtColor(Image,cv2.COLOR_BGR2GRAY)

#binarization
ret,thresh = cv2.threshold(gray,135,255,cv2.THRESH_BINARY_INV)

#dilation
kernel = np.ones((2,100),np.uint8)
img_dilation=cv2.dilate(thresh,kernel,iterations=1)

#find contours
im2, ctrs, hier = cv2.findContours(img_dilation.copy(),cv2.
    RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)

for i,ctr in reversed(list(enumerate(ctrs))):

    #get bounding box
    x, y, w, h= cv2.boundingRect(ctr)

    #Getting roi
    roi=image[y:y+h , x:x+w]
    find_contour(roi,i)
```

Listing 4.2: Finding contours in script

```python
def find_contour(roi,i):
    I=roi.copy()
    G_Image=cv2.cvtColor(roi,cv2.COLOR_BGR2GRAY)

    th=cv2.adaptiveThreshold(G_Image,255,cv2.ADAPTIVE_THRESH_MEAN_C,
        cv2.THRESH_BINARY_INV,21,2)

    image, contours, hierarchy =cv2.findContours(th,cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)
    count=0
    e=hierarchy.shape[1]

    l=[]
    for contour in contours:
        if(cv2.contourArea(contour) >3):

            [x,y,w,h]=cv2.boundingRect(contour)

            M=cv2.moments(contour)
            if(M["m00"]!=0):
```

```
            cX=int(M["m10"] / M["m00"])
            cY=int(M["m01"] / M["m00"])
        else:
            cX, cY=0, 0
        l.append([x,y,w,h,cX])
```

After finding the middle coordinate of each bounding box, boxes were sorted to get the order of characters in the script like in following figure 4.3.

Listing 4.3: Sort the contours to get the order of characters

```
def sorting(l,I):
    d=0
    e=0

    q=sorted(l, key=itemgetter(4))

    for i in range(len(q)):
        x=q[i][0]
        y=q[i][1]
        w=q[i][2]
        h=q[i][3]

        #crop image
        crop_img = I[y:y+h, x:x+w]
        cv2.imwrite("/research -OCR/segmentation/real data/data/"+str(
            e)+"image"+str(d)+".jpg",crop_img)
        e=e+1
        d=d+1
```

4.3.2 Character segmentation using convex hull

As mentioned in the section 3.2.2, first image is binarized using adaptive threshold binarization technique. Then by using *findContour()* function in OpenCV, contours are obtained for each character. After finding contours, Convex Hull class in OpenCV library is used to find convex hull for each of the contours. In order to obtain the character images in left to right in the order, contours are sorted before applying the convexHull function. Implementation of this method is shown in listing 4.4.

In Sinhala scripts, when some characters such as 'ර'segment, character following this character also tend to segment because 'ර'characters' bounding box coordinates overlap with following characters' coordinates. Unlike above approach, this technique segments the characters around it boundary. Therefore, this approach solves practical issues like above. However, sometimes contour function incorrectly identify the intensity variation in some characters and give only a part of character .

Listing 4.4: Finding convex hull of characters

```
#image binarization
threshed_img =cv2.adaptiveThreshold(blur,255,cv2.
    ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY_INV,139,2)
#finding contours
image, contours, hier = cv2.findContours(threshed_img, cv2.
    RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)

#Black image to be  used to draw individual convex hull
black1 = np.zeros(img.shape,dtype=np.uint8)

#sorts contours left to right so the image comes in order
contours = sorted(contours, key=lambda ctr: cv2.boundingRect(ctr)[0])

d=0
for cnt in contours:
    hull = cv2.convexHull(cnt)

    img3 = img.copy()
    black2 = black1.copy()

    im=cv2.drawContours(black2, [hull], -1,255, -1)
    g2 = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    r, t2 = cv2.threshold(g2, 127, 255,cv2.THRESH_OTSU)

    masked = cv2.bitwise_and(img2, img2,mask=t2)
```

4.3.3 Horizontal and vertical projection profile-based segmentation

In this technique, vertical projection of Sinhala script is used to identify lines and horizontal profile projection is used to segment characters and words. As mentioned in the section 3.2.3, sum of the black pixels perpendicular to the x and y axis is obtained.

After getting sum of black pixels, lines are separated using number of separation between vertical projections of lines while words are segmented using that of horizontal projections.

Listing 4.5: Obtaining the lines in the script

```
%round off the values >0 to 1 and others to 0
lines = pH > 0;

%Detect rising edge and falling edge
%gives the difference
d = diff(lines);

%find the indexes of value higher than 0 that means 1
```

```
startingColumns = find(d>0);

%find the indexes of value less than o
endingColumns = find(d<0);

for k = 1 : n

      subImage{k} = BW(startingColumns(k):endingColumns(k) ,:);

      %sum of the columns in the line
      pHline{k} = mean(subImage{k},1);
```

Listing 4.6: Character segmentation from words

```
for i=1:length(startingRow)

    word{i} = subImage{k}(:,startingRow(i):endingRow(i));
    wordarr{count} = word{i};

    %sum of the columns in the line
    pHword{i} = sum(word{i},1);

    wordN = pHword{i} > 0;
    q = diff(wordN);
    starting = find(q>0);
    ending= find(q<0);
    r=length(starting)-1;

    for s = 1 : r
        % Get sub image of just one character...
        subim = word{i}(:, starting(s):ending(s));

        % Now process this subimage of a single character....
        baseFileName = sprintf('line%d_%d.jpg',x,e);

        fullFileName = fullfile(someFolder, baseFileName);
        subim=~subim
        imwrite(subim,fullFileName );
        e=e+1;
    end;
 end;
```

4.3.4 Character recognition models implementation

As mentioned in section 3.4.2, convolutional neural network has been used in this study to recognize printed Sinhala characters. Experiments are carried out using the winning CNN architectures of ImageNet Large Scale Visual Recognition Challenge.

1. AlexNet CNN Architecture [20]

AlexNet CNN architecture is implemented using Keras library and Tensorflow framework. In the AlexNet architecture they have used 224x224 size input images. However, in this case image size was reduced to 28x28 image size since experiments are done to character images. Kernel size and the stride size was reduced in a way that is suitable for the input image size. Furthermore, number of kernels in each layer was not changed and only the number of nodes in last layer was changed to 297, as there are 297 classes in the training data set.

Listing 4.7: AlexNet architecture

```
# 1st Convolutional Layer
model.add(Conv2D(filters=96, input_shape=(28,28,1),kernel_size=(7,7),\
 strides=(2,2), padding='valid'))
model.add(Activation('relu'))

# Pooling
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'
    ))

# Batch Normalisation before passing it to the next layer
model.add(BatchNormalization())

# 2nd Convolutional Layer
model.add(Conv2D(filters=256, kernel_size=(5,5), strides=(1,1),
    padding='valid'))
model.add(Activation('relu'))

# Pooling
model.add(MaxPooling2D(pool_size=(1,1), strides=(2,2), padding='valid'
    ))

# Batch Normalisation
model.add(BatchNormalization())

# 3rd Convolutional Layer
model.add(Conv2D(filters=384, kernel_size=(1,1), strides=(1,1),
    padding='valid'))
model.add(Activation('relu'))

# Batch Normalisation
model.add(BatchNormalization())

# 4th Convolutional Layer
model.add(Conv2D(filters=384, kernel_size=(1,1), strides=(1,1),
    padding='valid'))
model.add(Activation('relu'))

# Batch Normalisation
model.add(BatchNormalization())

# 5th Convolutional Layer
```

```
model.add(Conv2D(filters=256, kernel_size=(1,1), strides=(1,1),
    padding='valid'))
model.add(Activation('relu'))

# Pooling
model.add(MaxPooling2D(pool_size=(1,1), strides=(2,2), padding='valid'
    ))

# Batch Normalisation
model.add(BatchNormalization())

# Passing it to a dense layer
model.add(Flatten())

# 1st Dense Layer
model.add(Dense(4096, input_shape=(28,28,1)))
model.add(Activation('relu'))

# Add Dropout to prevent overfitting
model.add(Dropout(0.4))

# Batch Normalisation
model.add(BatchNormalization())

# 2nd Dense Layer
model.add(Dense(4096))
model.add(Activation('relu'))

# Add Dropout
model.add(Dropout(0.4))

# Batch Normalisation
model.add(BatchNormalization())

# 3rd Dense Layer
model.add(Dense(1000))
model.add(Activation('relu'))

# Add Dropout
model.add(Dropout(0.4))

# Batch Normalisation
model.add(BatchNormalization())

# Output Layer
model.add(Dense(284))
model.add(Activation('softmax'))
```

2. LeNet Architecture [21]

In the LeNet architecture proposed in the study [21], has used 32x32 pixel grayscale input images. However, in this scenario image size was reduced to 28x28 image size since experiments are done to character images. Kernel size and the stride size was reduced in a way that is suitable for the input image size. Furthermore, number of kernels in each layer was not changed and only the number of nodes in last layer was changed to 297, as there are 297 classes in the training data set.

Listing 4.8: LeNet architecture

```
# (3) Create a sequential model
model = Sequential()
model.add(Conv2D(filters = 6,
                 kernel_size = 5,
                 strides = 1,
                 activation = 'relu',
                 input_shape = (28,28,1)))
#Pooling layer 1
model.add(MaxPooling2D(pool_size = 2, strides = 2))
#Layer 2
#Conv Layer 2
model.add(Conv2D(filters = 16,
                 kernel_size = 5,
                 strides = 1,
                 activation = 'relu',
                 input_shape = (14,14,6)))

#Pooling Layer 2
model.add(MaxPooling2D(pool_size = 2, strides = 2))
#Flatten
model.add(Flatten())

#Layer 4
#Fully connected layer 2
model.add(Dense(units = 120, activation = 'relu'))
#Layer 5
model.add(Dense(units = 84, activation = 'relu'))
#Output Layer
model.add(Dense(units =293, activation = 'softmax'))
```

3. ResNet Architecture

ResNet architecture is an architecture with deeper layers upto 152. When trained this model default input size of images were changed to 28 x 28 and the number of nodes in outer dense layers was changed to the number of classes in the data set. However, when compare to the other two models mentioned above, ResNet model took more time to train.

Listing 4.9: ResNet architecture

```
 x = ZeroPadding2D((3, 3))(img_input)
x = Conv2D(64, (7, 7), strides=(2, 2), name='conv1')(x)
x = BatchNormalization(axis=bn_axis, name='bn_conv1')(x)
x = Activation('relu')(x)
x = MaxPooling2D((3, 3), strides=(2, 2))(x)

```

```
x = conv_block(x, 3, [64, 64, 256], stage=2, block='a', strides=(1, 1)
    )
x = identity_block(x, 3, [64, 64, 256], stage=2, block='b')
x = identity_block(x, 3, [64, 64, 256], stage=2, block='c')

x = conv_block(x, 3, [128, 128, 512], stage=3, block='a')
x = identity_block(x, 3, [128, 128, 512], stage=3, block='b')
x = identity_block(x, 3, [128, 128, 512], stage=3, block='c')
x = identity_block(x, 3, [128, 128, 512], stage=3, block='d')

x = conv_block(x, 3, [256, 256, 1024], stage=4, block='a')
x = identity_block(x, 3, [256, 256, 1024], stage=4, block='b')
x = identity_block(x, 3, [256, 256, 1024], stage=4, block='c')
x = identity_block(x, 3, [256, 256, 1024], stage=4, block='d')
x = identity_block(x, 3, [256, 256, 1024], stage=4, block='e')
x = identity_block(x, 3, [256, 256, 1024], stage=4, block='f')

x = conv_block(x, 3, [512, 512, 2048], stage=5, block='a')
x = identity_block(x, 3, [512, 512, 2048], stage=5, block='b')
x = identity_block(x, 3, [512, 512, 2048], stage=5, block='c')

x = AveragePooling2D((1,1), name='avg_pool')(x)
```

4.3.4.1 Implementation of the Final Model

As described in the section 5.3, AlexNet architecture provided the higher cross validation accuracy for the created data set. Hence, similar architecture to the AlexNet was used as the final model to recognize Sinhala printed characters. Models were trained using Google Colab [?] in order to run deep learning model on GPU, since it supports free GPU. Final model used to recognize Sinhala characters consisted with five convolutional layers and three dense layers where outer layer consisted of 297 nodes for the 297 classes. First, second and the fifth convolutional layer was followed by a MaxPooling layer and each of the convolutional layer followed by a batch normalization. Each dense layer is followed by a dropout of 0.4 probability. All the character images were resized into fixed size 28 x 28 before input into the CNN. Pixel values of the images were input into the CNN model. Batch size of 20 was used in training the final model since when the batch size was increased it caused a memory error in Google Colab. Adam optimizer is used in training the final model since it learns faster and stable than other optimizers as suggested in [27]. Default learning rate 0.001 for “adam” optimizer in Keras was used as the initial learning rate.

The predicted classes by the trained model was saved as a text file and predicted classes were mapped to the corresponding character classes. As the final step, Characters given by the mapper was normalized using a normalizer to get the final output.

Chapter 5

Results and evaluation

5.1 Introduction

This chapter discusses the results and evaluation of the proposed design under three sections. Section 5.2 explains the performance evaluation of character segmentation technique. In the section 5.3, character recognition performance of convolutional neural network is evaluated. Section 5.4 explains the overall performance of proposed model.

5.2 Character segmentation

5.2.1 Character segmentation by using contours

Character segmentation using contours technique has been experimented using font size 12 Sinhala scripts images with 300 DPI value and these images were created using different fonts. Font size 12 is chosen since it is the minimum font size consider in this study. Following table give information of the total number of characters in the script, number of correctly segmented characters and the sensitivity values for this segmentation technique in each image and for all images together.

Table 5.1: Accuracies for character segmentation using contour based segmentation

Image	No. of characters	No. of correctly segmented characters	Sensitivity
1	714	689	96.50%
2	1082	921	85.12%
3	732	683	93.31%
4	720	557	77.36%
5	1062	800	75.33%
6	720	715	99.31%
7	1002	887	88.52%
8	720	696	96.67%
9	1062	913	85.97%
10	714	557	78.01%
Total	**8528**	**7418**	**86.98%**

Overall accuracy of this character segmentation technique is 86.98%. Characters with '◌ු', '◌ූ', 'ෙ◌' modifiers and some font types has been affected the overall accuracy of this character segmentation technique. Even though a character in one font type correctly segmented, when the font type changes ability to segment the character has changed with font type.

5.2.2 Horizontal and vertical projection-based character segmentation

Scripts which has been used in character segmentation using contours, were used to evaluate the horizontal and vertical projection-based character segmentation. Following table shows the information of the total number of characters in the script, number of correctly segmented characters and the sensitivity values for this segmentation technique in each image and for all images together.

Table 5.2: Accuracies for horizontal and vertical projection-based character segmentation

Image	No. of characters	No. of correctly segmented characters	Sensitivity
1	714	677	94.82%
2	1082	948	87.62%
3	732	683	93.31%
4	720	553	76.81%
5	1062	728	68.55%
6	720	640	88.88%
7	1002	762	76.05%
8	720	433	60.14%
9	1062	488	45.95%
10	714	647	90.62%
Total	**8528**	**6559**	**76.91%**

As shown in the results of the two methods, contour-based character segmentation has achieved a higher accuracy of 86.98% in character segmentation in contrast to the accuracy of 76.91% obtained by the character segmentation using horizontal and vertical projection.

Character segmentation by using contours, has identified less number of unimportant regions and incorrectly segment characters with compare to the horizontal and vertical projection-based method for fonts such as “Iskolapota”, “Malithi Web”, “Nirmala” and “Sarasavi Unicode”. However, the font “BhashitaScreen” has been correctly segmented with less unimportant regions by horizontal and vertical projection-based method. The reason for segmenting unimportant regions other than characters is that these images were contained some noise and contours have drawn around these regions. Some characters were not properly segmented due to the fact that intensity variations around the characters were unable to identify by these methods. Even though some characters were properly segmented for one font type, when the font type changes ability to segment characters get changed. However, since the overall performance for contour-based character segmentation method is higher than other method, character segmentation using contour-based method was used as the character segmentation technique for the character recognition process.

5.3 Character recognition

5.3.1 Experimental setup

As mentioned in section 3.2.4, character recognition is experimented by using several CNN architectures in ImageNet Large Scale Visual Recognition Challenge such as AlexNet [20], LeNet [21], ResNet [23] and GoogleNet [22] . CNN models were trained using character images of font size 12, 14 and 16. Experiments were carried out using several font size combinations in order to address the second research question mentioned in section 1.2. Following table summarize the testing accuracy obtain for each CNN architecture using combination of font sizes 12, 14 and 16.

Table 5.3: Results of font size combinations for each architecture

Character Combination		AlexNet Architecture	LeNet Architecture	ResNet Architecture
Training font size	Testing font size			
12	12	98.06%	96.97%	91.31%
14	14	98.56%	98.46%	93.45%
16	16	98.86%	98.40%	96.34%
12, 14, 16	12,14,16	85.45%	79.04%	75.64%
12	12,14,16	58.57%	51.02%	46.55%
14	12,14,16	69.40%	63.23%	62.63%
16	12,14,16	58.84%	53.75%	52.23%
12,14,16	12	86.21%	76.89%	71.58%
12,14,16	14	87.71%	81.26%	78.34%
12,14,16	16	87.86%	78.41%	75.82%
12,14	12,14,16	65.43%	64.50%	63.56%
14,16	12,14,16	70.47%	68.45%	65.43%
12,16	12,14,16	71.32%	71.20%	69.63%
12,14,16	12,14	79.84%	78.67%	70.60%
12,14,16	14,16	80.82%	79.34%	78.66%
12,14,16	12,16	75.68%	73.63%	71.48%

As shown in the test accuracy rates for each CNN architecture, higher test accuracy rates were given when the CNN models were trained using character combination of font size 12, 14, and 16 and the test set consist with any of the font size out of font sizes 12,14, and 16. Data to train, test and validate each model were chosen randomly and due to this fact testing accuracy get varied. Therefore, average accuracy was obtained by training and testing the model several times for each font size combination. Thus, the final model presented in this study was trained using the character combination of font size 12, 14, and 16.

The prediction accuracy for each architecture was evaluated using 5-fold cross validation. For each architecture, cross validation accuracy are shown in Table 5.4.

Table 5.4: Cross validation accuracy for each CNN architecture

Architecture	Cross Validation Accuracy
AlexNet	95.92% (+/- 0.87%)
LeNet	93.72% (+/- 0.77%)
ResNet	92.32% (+/- 0.74%)

Since AlexNet architecture has higher cross validation accuracy with contrast to other architectures, a CNN architecture similar to the AlexNet was used as the final model.

5.3.2 Character recognition performance

Character recognition performance was evaluated underneath two conditions; character recognition of characters segmented automatically and manually, character recognition of accurately segmented characters and the unimportant segments.

1.Recognition of characters segmented using contour-based method and manually segmented characters.

As mentioned in the section 5.2, characters segmented from the contour-based character segmentation method includes inaccurately segmented characters as well as the correctly segmented characters. Hence, in order to measure the Sinhala character recognition performance when apply deep learning approach, unsegmented characters were segment manually. For the all automatically segmented and manually segmented characters in each script, character recognition performance were evaluated.

Character recognition accuracy for each script is presented in following table 5.5. This recognition accuracy has obtained by manually segmenting the characters which were not properly segmented. Following table shows the number of characters in each script, number of correctly classified characters in each script and the classifier sensitivity.

Table 5.5: Accuracy of the automatically and manually segmented characters

Image	No. of characters	No. of correctly classified characters	Classifier Accuracy
1 - Script with ‘Iskoola Pota’ font	714	546	76.47%
2 - Script with ‘Iskoola Pota’ font	977	918	93.96%
3 - Script with ‘Nirmala‘ font	716	668	93.30%
4 - Scanned image	603	560	92.87%
5 - Scanned image	714	651	91.18%
Total	3724	3343	89.76%

In this study main focus is to present a deep learning approach to improve the character recognition of Sinhala characters. Therefore, the results obtained in this study has shown that using a deep learning approach accuracy of 89.76% could be obtained.

Important to mention the fact that this study has used all the Sinhala characters including the complex characters as the data set. Hence, when compared to the studies [6], [11] that have achieved accuracies between 80% - 90% range only by using a subset of Sinhala characters as the data set, the accuracy obtained using all the Sinhala characters including the complex characters, is notable.

Performance of the final model was evaluated using five scripts. Overall precision, recall and F1 measure values obtained for all the images of script is explained in following paragraphs.

Results evaluation

Matrices which was used in the evaluation are; precision, recall, F1-score. Precision represent the correct percentage of all the instances that classified as positive.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \qquad \text{(equation 5.1)}$$

Macro average is the average value of the precision or recall or f1-score of all the classes. Micro average is same as micro average however, it aggregates the contribution of all classes to compute the average value. Since this is a multi-class classification, micro average is preferable to present the precision and recall value of the classifier in evaluating the characters in test images as the images may have class imbalance. Micro average is more dominant towards the most populated class and the macro average is more dominant toward the least populated classes.

Recall is the ability of a classifier to find all positive instances.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \qquad \text{(equation 5.2)}$$

F1-score is a weighted harmonic mean of precision and recall such that the best score is 1.0.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \qquad \text{(equation 5.3)}$$

Support column in the following tables shows the number of characters contain in each image from each class mention in the left most column.

Precision, recall, f1-score value of the classifier is shown in Figure A.7 to Figure A.10.

Overall precision, recall, and F1-score value

The classifier has given a 0.89 micro average of precision which means the classifier has performed well for the most populated classes in the test images. Since the macro average value also near to 1.0, can conclude that the classifier has performed better in classifying least populated classes. Eleven-character classes has given precision value equal or less than 0.5. Characters 'ක්‍ෂූ', 'බ', 'ශ','ඝී','ඳු','ඳු', 'ඳි' , 'හ' ,'ලු', 'ඕ','හ්','◌ෘ' has given precision value less than or equal to 0.5. However, for some characters even though the precision value is 1.0, recall value is less than or equal to 0.5, which means classifiers ability to identify all the relevant characters for each of these classes is low. Characters that has shown a low recall value are; 'ඝී', 'ලී', 'ඝි','ළී' , 'ළු' Overall, classes have given 0.89 micro average of recall value which is good since it is near to 1.0. F1-score measure which shows tradeoff between the precision and recall values has given a 0.89 micro average of f1-score which shows the realistic performance of this classifier is good since the value is near to 1.0.

2.Character recognition of accurately segmented characters and the unimportant segments

Character recognition performance of all the accurately segmented, unsegmented and unimportant segments were evaluated for the above-mentioned images. The table 5.6 shows the results of the recognition.

Table 5.6: Overall accuracy of the classifier including segmentation accuracy

Image	No.of segments	No.of accurate segments	No.of classified characters	Classifier Sensitivity	Overall Sensitivity
1	717	685	645	89.95%	85.94%
2	958	916	873	91.12%	87.22%
3	718	712	695	95.98%	93.34%
4	636	600	571	89.77%	83.78%
5	791	714	651	82.30%	76.89%
Total	3820	3627	3435	89.92%	85.37%

As mentioned in previous section 5.2, when images are gone through character segmentation process based on contours, it has segmented the characters accurately as well as there were unimportant segments due to the noise and intensity variations. According to the overall accuracy obtained when consider the segmentation accuracy and the recognition accuracy there were small reduction in the overall accuracy with compare to the accuracy obtained for the auto and manually segmented characters.

Character recognition performance according to the font size

Character recognition performance of proposed deep learning approach was evaluated according to the font size. Following figure shows the accuracy improvement obtained when the model was trained using the font size 12,14, and 16 and test with only the 12px, 14px and 16px. The results of the experiment show that when the font size increased, character recognition performance has increased. Another experiment was carried out to test the impact of size of the training data set in character recognition performance. The test dataset used in both these experiments were same. The size of the two-training dataset was 15,884 and 48,472 characters respectively. The results have been presented that when the training data set has increased, character recognition performance has been increased. Therefore, this result has proved that more the data accuracy gets better. In Figure 5.1, dataset 1 and dataset 2 corresponding to the dataset consisted with 15,884 characters and 48, 472 respectively.

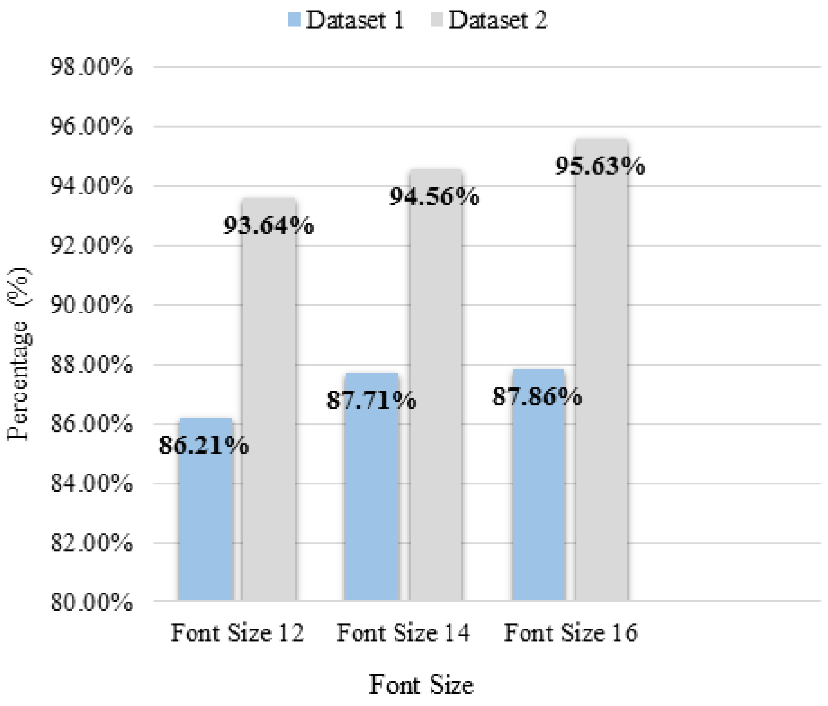


Figure 5.1: Sensitivity value of character recognition according to the font size

Error analysis of test images

According to the confusion matrix (Figure A.6: Confusion Matrix obtained for the all test images), two characters were completely misclassified by the classifier. Character ‘ඩෙ’ has completely misclassified as character ‘ා’ and character ‘ළු’ has completely misclassified as character ‘ෙ◌’ . In test images characters such as ‘ෘ’, ‘හ’, ‘චි’,‘ස’, ‘ම්’, ‘ඔ’,‘ක්‍ෂු’,‘බු’ were misclassified as ‘ෟ’, ‘හී’, ‘චි’,‘ස’ , ‘මී’, ‘ම’,‘ක්‍ෂු’,‘බු’ respectively. However, in overall prediction results these characters have high probability to classify as true character, except the characters ‘ෘ’ and ‘ෟ’ since each of these two characters have 0.5 predictive result.

Following Figure 5.2 shows the final outcome of the image 4 after mapping the predicted classes to the corresponding Sinhala character. (Output of other four images of scripts are listed in Appendix A)

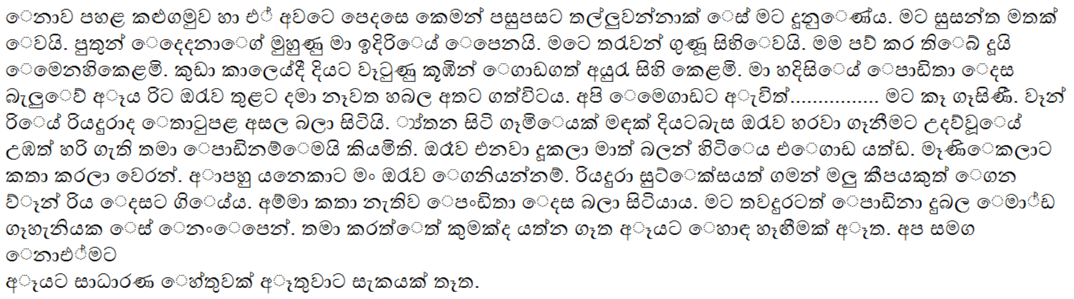

ෙනාව පහළ කළුගමුව හා එ් අවටෙ පෙදසෙ කෙමන් පසුපසට තල්ලුවන්නාක් ෙස් මට දුනුෙණ්ය. මට සුසන්ත මතක් ෙවයි. පුතුන් ෙදෙදනාෙග් මුහුණු මා ඉදිරිෙය් ෙපෙනයි. මටෙ තරුවන් ගුණු සිහිෙවයි. මම පවි කර තිෙබ් දුයි ෙමෙනහිකෙළමි. කුඩා කාලෙය්දී දියට වැටුණු කූඹින් ෙගාඩගත් අයුරු සිහි කෙළමි. මා හදිසිෙය් ෙපාඩිතා ෙදස බැලුෙව් අැය රිට ඔරුව තුළට දමා නැවත හබල අතට ගත්විටය. අපි ෙමෙගාඩට අැවිත්............... මට කෑ ගැසිණි. වැන් රිෙය් රියදුරාද ෙතාටුපළ අසල බලා සිටියි. ්තන සිටි ගෑමිෙයක් මඳක් දියටබැස ඔරුව හරවා ගැනීමට උදව්වූෙය් උඹත් හරි ගැති තමා ෙපාඩිනම්ෙමයි කියමිති. ඔරුව එනවා දූකලා මාත් බලන් හිටිෙය එෙගාඩ යත්ඩ. මැණිෙකලාට කතා කරලා වෙරන්. අාපහු යනෙකාට මං ඔරුව ෙගනියන්නම්. රියදුරා සුට්ෙක්සයත් ගමන් මලු කීපයකුත් ෙගන ව්ැන් රිය ෙදසට ගිෙය්ය. අම්මා කතා නැතිව ෙපංඩිතා ෙදස බලා සිටියාය. මට තවදුරටත් ෙපාඩිනා දුබල ෙමා්ඩ ගැහැනියක ෙස් ෙනංෙපෙන්. තමා කරත්ෙත් කුමක්ද යත්න ගැත අැයට ෙහාඳ හැඟීමක් අැත. අප සමග
ෙනාඑ්මට
අැයට සාධාරණ ෙහ්තුවක් අැතුවාට සැකයක් තැත.

Figure 5.2: Output after mapping predicted classes for the image 4

5.3.3 Summary

Section 5.2 have explained the evaluation of character segmentation techniques and the pros and cons in each approach. The better approach has obtained by evaluating and it has used in final proposed solution.

In the section 5.3, results of character recognition models were discussed and final model has been suggested using the cross-validation accuracy of each model. Character recognition performance was evaluated underneath two criteria; Assuming 100% segmentation accuracy in segmentation step, Considering the real segmentation accuracy. Character recognition process assuming 100% segmentation has been given 89.76% accuracy for the character recognition using a CNN and overall 85.37% has achieved including the segmentation accuracy.

Chapter 6

Conclusions

6.1 Introduction

This chapter includes a review on research aims and objectives, research problem, limitations of the current work and implications for further research.

6.2 Conclusions about research questions

The main aim of this project is to explore the applicability of deep learning approach to improve Sinhala optical character recognition. A convolutional neural network was investigated for Sinhala OCR.

Creation of character data set by using Sinhala printed characters is an objective in this study. Hence, as the first step, training character set was created by converting screen text into jpeg images.

One of the research questions in this study was to identify the effect of character segmentation technique for recognising characters. Thus, three-character segmentation techniques were experimented and the better approach was chosen to segment characters. Furthermore, to explore the effect of contour-based segmentation method in recognizing characters, recognition process was conducted in two approaches. First approach considered the well segmented characters (unsegmented ones were manually segmented) after applying contour-based segmentation. Second approach has considered all the segments that were obtained applying segmentation method. According to the results obtained for two approaches, first approach has been depicted better results than the second approach. That is to say character segmentation method has an impact on character recognition process as the first approach consisted with only well segmented characters. Hence, improvement in character segmentation step could be given an increment in the performance of the character recognition.

The effect of the training font size was evaluated by conducting several experiments mentioned in the section 5.3. The results have exposed that by using the combination of font size 12, 14, and 16 better accuracy could be obtained rather than when using single font size for training. According to the results obtained from these experiments, when the font size of the test set was increased, the recognition accuracy was improved. The results have depicted that by increasing the size of the training data set, increment of the accuracy could be obtained as mentioned in a previous study in section 1.3.

As mentioned in section 1.3, previous studies [3], [6] have mentioned the character recognition accuracy between 85%- 90% range. However, these studies have considered an approach to recognize only a subset of Sinhala characters, mainly the core characters. However, approach suggested by this study, has been considered all the Sinhala characters that are used in Sinhala writings. Therefore, overall accuracy 85.37%, obtained including the character segmentation step accuracy, is quite significant with contrast to the previous studies mentioned in the chapter 2. Thus, it can be concluded that the proposed approach can be utilized to improve the character recognition of Sinhala printed characters over the existing approaches that was considered only a subset of Sinhala characters.

6.3 Conclusions about research problem

The identification of different font sizes in Sinhala optical character recognition can be improved by using a convolutional neural network that has been trained using different font sizes. According to the confusion matrix in Figure A.6, obtained for the tested images, character ‘ක’ has shown correctly predicted value of 0.91 which means the classifier has predicted the character accurately most of the time. Like character ‘ක’ character ‘ත’ has also shown correctly predicted value of 0.91. Character ‘ට’ and ‘ච’ has given 0.99 and 0.95 correctly predicted values respectively. Consequently, the values of the diagonal elements represent the correctly predicted classes. The confusion is expressed by the false classified off-diagonal elements, since they are mistakenly confused with another class. Hence, the proposed approach has achieved the ability to differentiate similar characters to some extent.

The proposed deep learning approach is capable of learning the representation features from the input images unlike hand-crafted features. As many other machine learning studies, images of Sinhala character data set have not publicly available for research purposes. Therefore, in addition to above mentioned contribution, another contribution in this study is creating an image data set for Sinhala character recognition.

6.4 Limitations

The character segmentation techniques used in this study showcased that even two or more scripts get scanned in same conditions, character segmentation gives different segmentation accuracy according to the font type used in the script. Hence, the character segmentation accuracy can be varied due to its dependency to segmentation accuracy. The output obtained from this study depicted that layout or the alignments of the text is not preserved in the final output. Also, the characters could not be output according to the character order although it identified by the model accurately, since the characters was interchanged when printing the output into a text file.

6.5 Implications for Further Research

Ensemble method with CNN and Long-Short Term Memory (LSTM) architecture can be investigated to improve the accuracy of Sinhala OCR. In this scenario, CNN could be used for feature extraction while a LSTM could be used for recognition purposes.

References

[1] R. Weerasinghe, A. Wasala, D. Herath, and V. Welgama, "Nlp applications of sinhala: Tts & ocr," in *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*, 2008.

[2] H. L. Premaratne, "Recognition of printed sinhala characters by direction fields," Ph.D. dissertation, Chalmers tekniska högskola, 2005.

[3] S. Ajward, N. Jayasundara, S. Madushika, and R. Ragel, "Converting printed sinhala documents to formatted editable text," in *Information and Automation for Sustainability (ICIAFs), 2010 5th International Conference on.* IEEE, 2010, pp. 138–143.

[4] S. Hewavitharana and N. Kodikara, "A statistical approach to sinhala handwriting recognition," in *Proc. of the International Information Technology Conference (IITC), Colombo, Sri Lanka*, 2002.

[5] G. Gunarathna, M. Chamikara, and R. Ragel, "A fuzzy based model to identify printed sinhala characters," in *Information and Automation for Sustainability (ICIAfS), 2014 7th International Conference on.* IEEE, 2014, pp. 1–6.

[6] M. Chamikara, S. Kodituwakku, A. Jayathilake, and K. Wijeweera, "Fuzzy neural hybrid method for sinhala character recognition," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 9, pp. 8–21, 2014.

[7] K. Karunarathne, K. Liyanage, D. Ruwanmini, G. Dias, and S. Nandasara, "Recognizing ancient sinhala inscription characters using neural network technologies," *Internationa Journal of Scientific Emgineering and Applied Sciences*, vol. 3.

[8] ""subasa"," http://www.subasa.lk/aocr/ocr.php, (Accessed on 01/29/2019).

[9] S. Prabhanjan and R. Dinesh, "Deep learning approach for devanagari script recognition," *International Journal of Image and Graphics*, vol. 17, no. 03, p. 1750016, 2017.

[10] M. M. R. Sazal, S. K. Biswas, M. F. Amin, and K. Murase, "Bangla handwritten character recognition using deep belief network," in *Electrical Information and Communication Technology (EICT), 2013 International Conference on*. IEEE, 2014, pp. 1–5.

[11] S. Hewavitharana, H. Fernando, and N. Kodikara, "Off-line sinhala handwriting recognition using hidden markov models." in *ICVGIP*, 2002.

[12] M. Karunanayaka, C. A. Marasinghe, and N. Kodikara, "Thresholding, noise reduction and skew correction of sinhala handwritten words." in *MVA*, 2005, pp. 355–358.

[13] "Opencv: Contours : Getting started," https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html, (Accessed on 01/30/2019).

[14] C. Silva and C. Kariyawasam, "Segmenting sinhala handwritten characters," *International Journal of Conceptions on Computing and Information Technology*, vol. 2, no. 4, pp. 22–26, 2014.

[15] H. L. Premaratne and J. Bigun, "A segmentation-free approach to recognise printed sinhala script using linear symmetry," *Pattern recognition*, vol. 37, no. 10, pp. 2081–2089, 2004.

[16] R. Anil, K. Manjusha, S. S. Kumar, and K. Soman, "Convolutional neural networks for the recognition of malayalam characters," in *Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014*. Springer, 2015, pp. 493–500.

[17] B. Balci, D. Saadati, and D. Shiferaw, "Handwritten text recognition using deep learning," *CS231n: Convolutional Neural Networks for Visual Recognition, Stanford University, Course Project Report, Spring*, 2017.

[18] Y. Shkarupa, R. Mencis, and M. Sabatelli, "Offline handwriting recognition using lstm recurrent neural networks," in *The 28th Benelux Conference on Artificial Intelligence*, 2016.

[19] T. Bluche, H. Ney, and C. Kermorvant, “Feature extraction with convolutional neural networks for handwritten word recognition,” in *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on.* IEEE, 2013, pp. 285–289.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[21] 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.

[22] 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, pp. 1–9.

[23] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[24] M. A. et al., “Tensorflow large-scale machine learning on heterogeneous systems,” https://www.tensorflow.org/, (Accessed on 01/29/2019).

[25] “Home - keras documentation,” https://keras.io/, (Accessed on 01/29/2019).

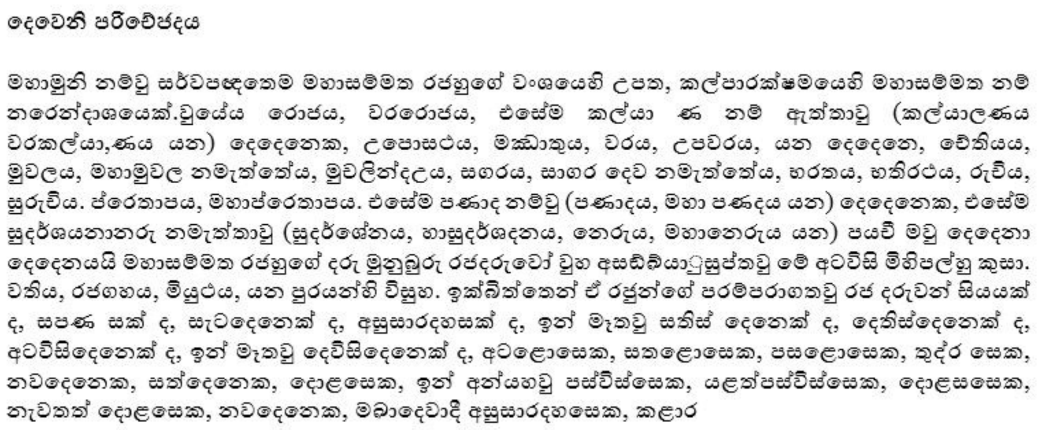
[26] “Google colaboratory,” https://colab.research.google.com/notebooks/welcome.ipynb, (Accessed on 01/29/2019).

[27] “How to pick the best learning rate for your machine learning project,” https://medium.freecodecamp.org/how-to-pick-the-best-learning-rate-for-your-machine-learning-project-9c28865039a8, (Accessed on 01/29/2019).

Appendix A

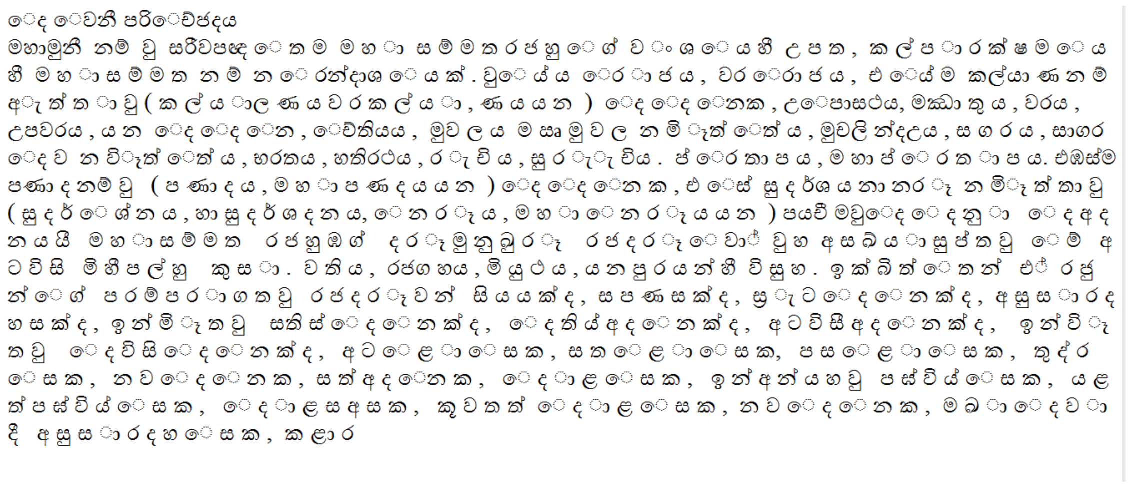
Results

Figure A.1 – A.5 shows the test images that was used in evaluating the proposed deep learning approach and the output of each test image after mapping to the correspondent character.



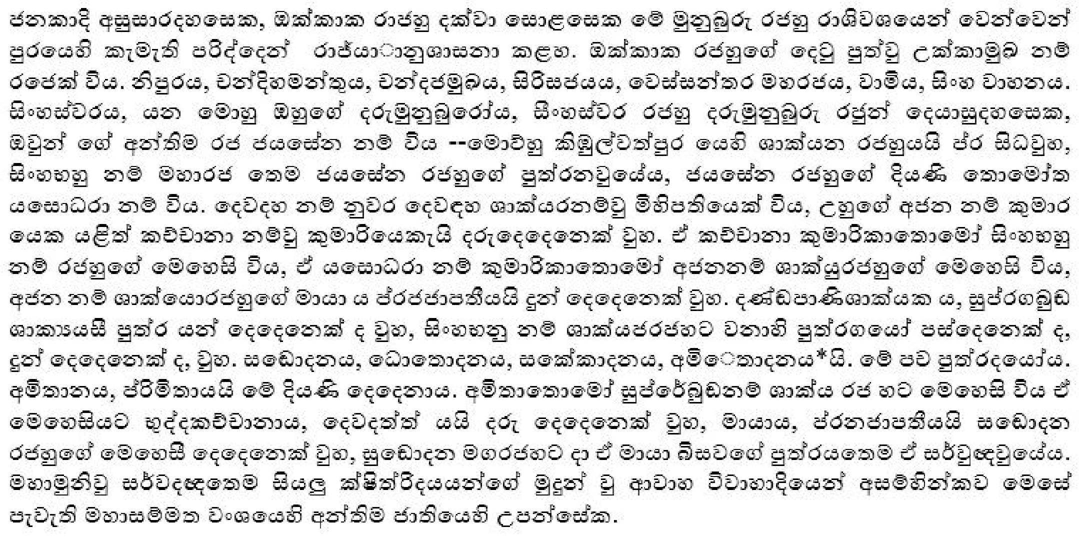
දෙවෙනි පරිච්ඡේදය

මහාමුනි නම්වූ සර්වපඥතෙම මහාසම්මත රජහුගේ වංශයෙහි උපත, කල්පාරක්ෂමයෙහි මහාසම්මත නම් නාරෙන්දාශයෙක්.වුයේය රොජය, වරරොජය, එසේම කල්යා ණ නම් ඇත්තාවූ (කල්යාලණය වරකල්යා,ණය යන) දෙදෙනෙක, උපොසථය, මඣාතුය, වරය, උපවරය, යන දෙදෙනෙ, චේතියය, මුචලය, මහාමුචල නමැත්තේය, මුචලින්දඋය, සගරය, සාගර දෙව නමැත්තේය, භරතය, භගිරථය, රුචිය, සුරුචිය. ප්රෙතාපය, මහාප්රෙතාපය. එසේම පණාද නම්වූ (පණාදය, මහා පණදය යන) දෙදෙනෙක, එසේම සුදර්ශයනානරු නමැත්තාවූ (සුදර්ශේනය, හාසුදර්ශදනය, නෙරුය, මහානෙරුය යන) පයචී මවූ දෙදෙනා දෙදෙනයයි මහාසම්මත රජහුගේ දරු මුනුබුරු රජදරුවෝ වූහ අසඞ්ඛ්යාු සුප්තවූ මේ අටවිසි මිහිපල්හු කුසා. වතිය, රජගහය, මියුථය, යන පුරයන්හි විසූහ. ඉක්බිත්තෙන් ඒ රජුන්ගේ පරම්පරාගතවූ රජ දරුවන් සියයක් ද, සපණ සක් ද, සැටදෙනෙක් ද, අසූසාරදහසක් ද, ඉන් මැතවූ සතිස් දෙනෙක් ද, දෙතිස්දෙනෙක් ද, අටවිසිදෙනෙක් ද, ඉන් මැතවූ දෙවිසිදෙනෙක් ද, අටළොසෙක, සතළොසෙක, පසළොසෙක, තුද්ර සෙක, නවදෙනෙක, සත්දෙනෙක, දොළසෙක, ඉන් අන්යහවූ පස්විස්සෙක, යළත්පස්විස්සෙක, දොළසසෙක, නැවතත් දොළසෙක, නවදෙනෙක, මඛාදෙවාදී අසූසාරදහසෙක, කළාර

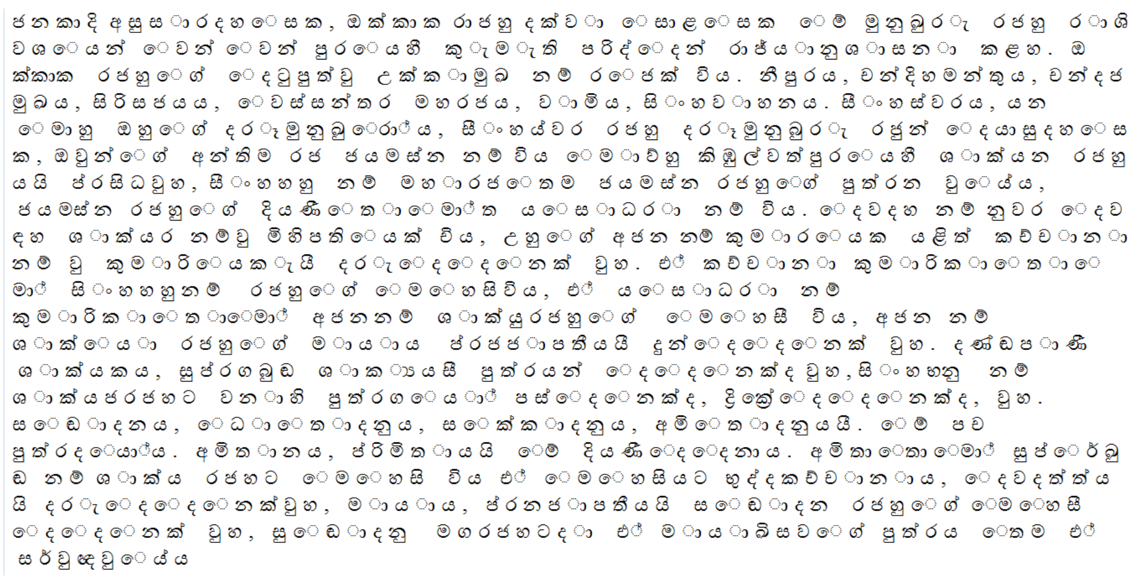


ෙද ෙවනී පරිෙච්ජදය
මහාමුනී නම් වූ සරීවපඥ ෙ ත ම ම හ ා ස ම් ම ත ර ජ හු ෙ ග් ව ං ශ ෙ ය හී උ ප ත , ක ල් ප ා ර ක් ෂ ම ෙ ය
හී ම හ ා ස ම් ම ත න ම් න ෙ රන්දාශ ෙ ය ක් . වු ෙ ය් ය ෙර ා ජ ය , වර ෙරා ජ ය , එ ෙය් ම කල්යා ණ න ම්
අ ැ ත් ත ා වූ (ක ල් ය ාල ණ ය ව ර ක ල් ය ා , ණ ය ය න) ෙද ෙද ෙනක , උ ෙපාසථය, මඣා තු ය , වරය ,
උපවරය , ය න ෙද ෙද ෙන , ෙචිතියය , මුච ල ය ම සෘ මු ච ල න මි ැත් ෙත් ය , මුචලි න්දඋය , ස ග ර ය , සාගර
ෙද ව න වි ැත් ෙත් ය , භරතය , භගිරථය , ර ැ චි ය , සු ර ැ ැ චිය . ප් ෙර තා ප ය , ම හා ප් ෙ ර ත ා ප ය. එඹස්ම
පණා ද නම් වූ (ප ණා ද ය , ම හ ා ප ණ ද ය ය න) ෙද ෙද ෙන ක , එ ෙස් සු ද ර්ශ ය නා නර ැ න මි ැ ත් තා වූ
(සු ද ර් ෙ ශ් න ය , හා සු ද ර් ශ ද න ය, ෙ න ර ැ ය , ම හ ා ෙ න ර ැ ය ය න) පයචී මවූෙද ෙ ද නු ා ෙ ද අ ද
න ය යී ම හ ා ස ම් ම ත ර ජ හු ඹ ග් ද ර ැ මු නු බු ර ැ ර ජ ද ර ැ ෙ වා ් වූ හ අ ස ඛ ය ා සු ප් ත වූ ෙ ම් අ
ට වි සි මි හී ප ල් හු කු ස ා . ව ති ය , රජග හය , මි යු ථ ය , ය න පු ර ය න් හී වි සූ හ . ඉ ක් බි ත් ෙ ත න් එ ් ර ජු
න් ෙ ග් ප ර ම් ප ර ා ග ත වූ ර ජ ද ර ැ ව න් සි ය ය ක් ද , ස ප ණ ස ක් ද , ශු ැ ට ෙ ද ෙ න ක් ද , අ සු ස ා ර ද
හ ස ක් ද , ඉ න් මි ැ ත වූ ස ති ස් ෙ ද ෙ න ක් ද , ෙ ද ති ය් අ ද ෙ න ක් ද , අ ට වි සී අ ද ෙ න ක් ද , ඉ න් වි ැ
ත වූ ෙ ද වි සි ෙ ද ෙ න ක් ද , අ ට ෙ ළ ා ෙ ස ක , ස ත ෙ ළ ා ෙ ස ක, ප ස ෙ ළ ා ෙ ස ක , තු ද් ර
ෙ ස ක , න ව ෙ ද ෙ න ක , ස ත් අ ද ෙන ක , ෙ ද ා ළ ෙ ස ක , ඉ න් අ න් ය භ වූ ප ස් වි ය් ෙ ස ක , ය ළ
ත් ප ස් වි ය් ෙ ස ක , ෙ ද ා ළ ස අ ස ක , කූ ව ත ත් ෙ ද ා ළ ෙ ස ක , න ව ෙ ද ෙ න ක , ම ඛ ා ෙ ද ව ා
දී අ සු ස ා ර ද භ ෙ ස ක , ක ළා ර

Figure A.1: Test Image 1 (script with font size 12 “Iskoola Pota” font)

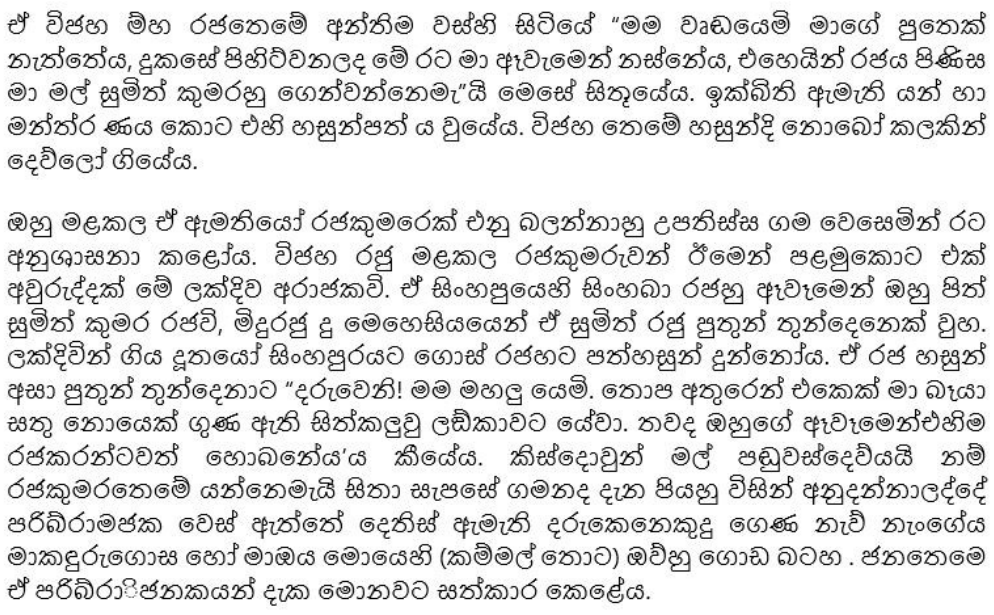


ජනකාදි අසුසාරදහසෙක, ඔක්කාක රාජහු දක්වා සොළසෙක මේ මුනුබුරු රජහු රාශිවශයෙන් වෙන්වෙන් පුරයෙහි කැමැති පරිද්දෙන් රාජ්යා◌ානුශාසනා කළහ. ඔක්කාක රජහුගේ දෙටු පුත්වූ උක්කාමුඛ නම් රජෙක් විය. නිපුරය, චන්දිහමන්තුය, චන්දජමුඛය, සිරිසජයය, වෙස්සන්තර මහරජය, වාමිය, සිංහ වාහනය. සිංහස්වරය, යන මොහු ඔහුගේ දරුමුනුබුරෝය, සිංහස්වර රජහු දරුමුනුබුරු රජුන් දෙයාසුදහසෙක, ඔවුන් ගේ අන්තිම රජ ජයසේන නම් විය --මොච්හු කිඹුල්වත්පුර යෙහි ශාක්යන රජහුයයි ප්ර සිධවුහ, සිංහහනු නම් මහාරජ තෙම ජයසේන රජහුගේ පුත්රනවූයේය, ජයසේන රජහුගේ දියණී තොමෝත යසොධරා නම් විය. දෙවදහ නම් නුවර දෙවඳහ ශාක්යරනම්වූ මීහිපතියෙක් විය, උහුගේ අජන නම් කුමාර යෙක යළිත් කච්චානා නම්වූ කුමාරියෙකැයි දරුදෙදෙනෙක් වූහ. ඒ කච්චානා කුමාරිකාතොමෝ සිංහහනු නම් රජහුගේ මෙහෙසි විය, ඒ යසොධරා නම් කුමාරිකාතොමෝ අජනනම් ශාක්යුරජහුගේ මෙහෙසි විය, අජන නම් ශාක්යොරජහුගේ මායා ය ප්රජජාපතීයයි දූන් දෙදෙනෙක් වූහ. දණ්ඩපාණීශාක්යක ය, සුප්රගබුඩ ශාකායයසී පුත්ර යන් දෙදෙනෙක් ද වූහ, සිංහහනු නම් ශාක්යජරජහට වනාහි පුත්රගයෝ පස්දෙනෙක් ද, දූන් දෙදෙනෙක් ද, වූහ. සඬොදනය, ධොතොදනය, සක්කොදනය, අමි◌ොතාදනය*යි. මේ පච පුත්රදයෝය. අමිතානය, ප්රිමිතායයි මේ දියණී දෙදෙනාය. අමිතාතොමෝ සුප්රේබුඩනම් ශාක්ය රජ හට මෙහෙසි විය ඒ මෙහෙසියට භුද්දකච්චානාය, දෙවදත්ත් යයි දරු දෙදෙනෙක් වූහ, මායාය, ප්රනජාපතීයයි සඬොදන රජහුගේ මෙහෙසී දෙදෙනෙක් වූහ, සුඬොදන මගරජහට දා ඒ මායා බිසවගේ පුත්රයතෙම ඒ සර්වුඥවූයේය. මහාමුනිවූ සර්වදඥතෙම සියලු ක්ෂිත්රිදයයන්ගේ මුදුන් වූ ආවාහ විවාහාදියෙන් අසම්භින්කව මෙසේ පැවැති මහාසම්මත වංශයෙහි අන්තිම ජාතියෙහි උපන්සේක.



ජ න කා දි අ සු ස ා ර ද හ ෙ ස ක , ඔ ක් කා ක රා ජ හු ද ක් ව ා ෙ සා ළ ෙ ස ක ෙ ම් මු නු බු ර ැ ර ජ හු ර ා ශි
ව ශ ෙ ය න් ෙ ව න් ෙ ව න් පු ර ෙ ය හි කු ැ ම ැ ති ප රි ද් ෙ ද න් රා ජ් ය ා නු ශ ා ස න ා ක ළ හ . ඔ
ක්කාක ර ජ හු ෙ ග් ෙ ද ටු පු ත් වූ උ ක් ක ා මු ඛ න ම් ර ෙ ජ ක් වි ය . නි පු ර ය , ච න් දි හ ම න් තු ය , ච න් ද ජ
මු ඛ ය , සි රි ස ජ ය ය , ෙ ව ස් ස න් ත ර ම හ ර ජ ය , ව ා මි ය , සි ං හ ව ා හ න ය . සී ං හ ස් ව ර ය , ය න
ෙ මා හු ඔ හු ෙ ග් ද ර ැ මු නු බු ෙරා ් ය , සී ං හ ය් ව ර ර ජ හු ද ර ැ මු නු බු ර ැ ර ජු න් ෙ ද යා සු ද හ ෙ ස
ක , ඔ වු න් ෙ ග් අ න් ති ම ර ජ ජ ය ම ස් න න ම් වි ය ෙ ම ා ච් හු කි ඹු ල් ව ත් පු ර ෙ ය හී ශ ා ක් ය න ර ජ හු
ය යි ප් ර සි ධ වු හ , සී ං හ හ හු න ම් ම හ ා ර ජ ෙ ත ම ජ ය ම ස් න ර ජ හු ෙග් පු ත් ර න වු ෙ ය් ය ,
ජ ය මස් න ර ජ හු ෙ ග් දි ය ණී ෙ ත ා ෙ මා ් ත ය ෙ ස ා ධ ර ා න ම් වි ය . ෙ ද ව ද හ න ම් නු ව ර ෙ ද ව
ඳ හ ශ ා ක් ය ර න ම් වු මී හි ප ති ෙ ය ක් වි ය , උ හු ෙ ග් අ ජ න න ම් කු ම ා ර ෙ ය ක ය ළි ත් ක ච් ච ා න ා
න ම් වු කු ම ා රි ෙ ය ක ැ යි ද ර ැ ෙ ද ෙ ද ෙ න ක් වු හ . එ ් ක ච් ච ා න ා කු ම ා රි ක ා ෙ ත ා ෙ
මා ් සි ං හ හ හු න ම් ර ජ හු ෙ ග් ෙ ම ෙ හ සි වි ය , එ ් ය ෙ ස ා ධ ර ා න ම්
කු ම ා රි ක ා ෙ ත ාෙමා ් අ ජ න න ම් ශ ා ක් යු ර ජ හු ෙ ග් ෙ ම ෙ හ සී වි ය , අ ජ න න ම්
ශ ා ක් ෙ ය ා ර ජ හු ෙ ග් ම ා ය ා ය ප් ර ජ ජ ා ප තී ය යි දු න් ෙ ද ෙ ද ෙ න ක් වු හ . ද ණ් ඩ ප ා ණී
ශ ා ක් ය ක ය , සු ප් ර ග බු ඩ ශ ා ක ාය යි පු ත් ර ය න් ෙ ද ෙ ද ෙ න ක් ද වු හ , සි ං හ හනු න ම්
ශ ා ක් ය ජ ර ජ හ ට ව න ා හි පු ත් ර ග ෙ ය ා ් ප ස් ෙ ද ෙ න ක් ද , දූ ක්‍රෝ ෙ ද ෙ ද ෙ න ක් ද , වු හ .
ස ෙ ඬ ා ද න ය , ෙ ධ ා ෙ ත ා ද නු ය , ස ෙ ක් ක ා ද නු ය , අ මි ෙ ත ා ද නු ය යි . ෙ ම් ප ච
පු ත් ර ද ෙයා ් ය . අ මි ත ා න ය , ප් රි මි ත ා ය යි ෙම් දි ය ණී ෙද ෙද නා ය . අ මි තා ෙතා ෙමා ් සු ප් ෙ ර් බු
ඩ න ම් ශ ා ක් ය ර ජ හ ට ෙ ම ෙ හ සි වි ය එ ් ෙ ම ෙ හ සි ය ට භු ද් ද ක ච් ච ා න ා ය , ෙ ද ව ද ත් ත් ය
යි ද ර ැ ෙ ද ෙ ද ෙ න ක් වු හ , ම ා ය ා ය , ප් ර න ජ ා ප තී ය යි ස ෙ ඬ ා ද න ර ජ හු ෙ ග් ෙම ෙහ සී
ෙ ද ෙ ද ෙ න ක් වු හ , සු ෙ ඬ ා ද නු ම ග ර ජ හ ට ද ා එ ් ම ා ය ා බි ස ව ෙ ග් පු ත් ර ය ෙත ම එ ්
ස ර් වු ඥ වු ෙ ය් ය

Figure A.2: Test Image 2 (script with font size 12 “Iskoola Pota” font)

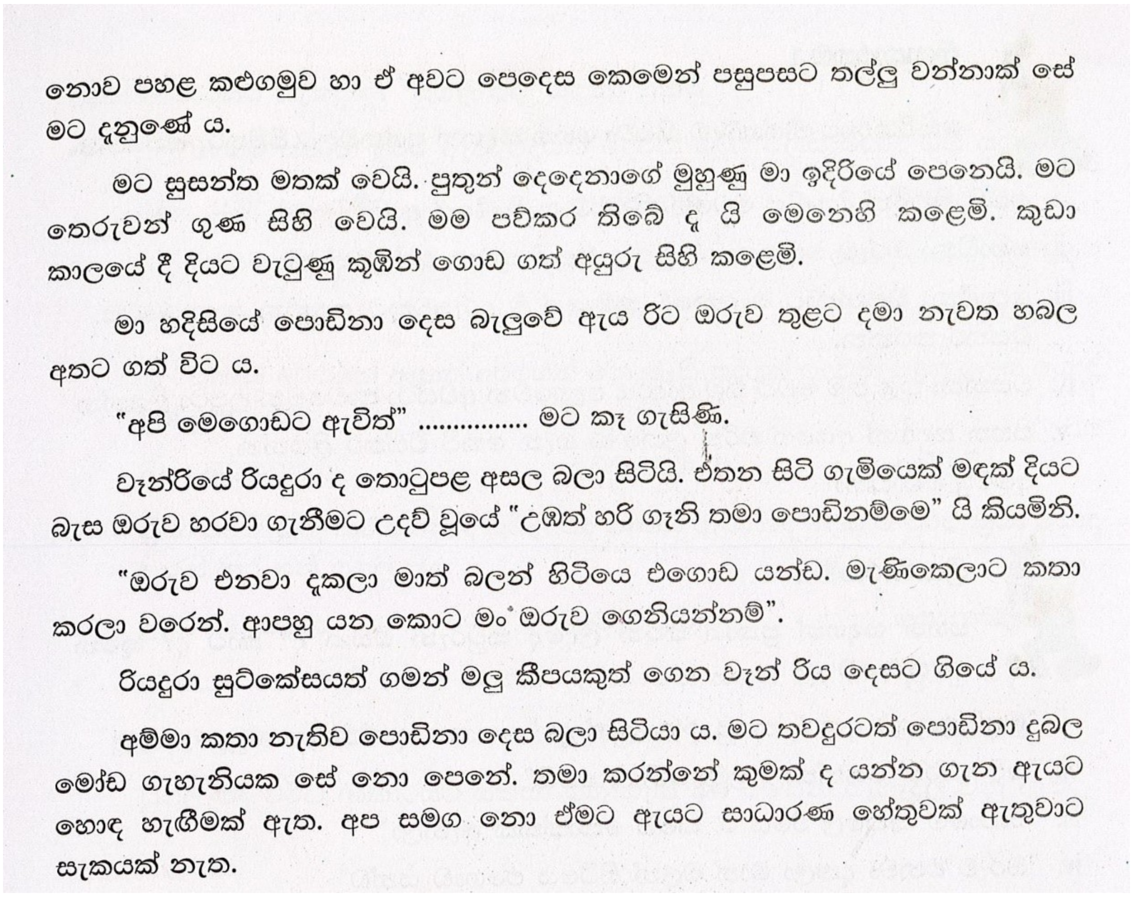


ඒ විජහ මිහ රජතෙමේ අන්තිම වස්හි සිටියේ "මම වෘඩයෙමි මාගේ පුතෙක් නැත්තේය, දුකසේ පිහිට්වනලද මේ රට මා ඇවෑමෙන් නස්නේය, එහෙයින් රජය පිණිස මා මල් සුමිත් කුමරහු ගෙන්වන්නෙමැ"යි මෙසේ සිතූයේය. ඉක්බිති ඇමැති යන් හා මන්ත්‍ර ණය කොට එහි හසුන්පත් ය වුයේය. විජහ තෙමේ හසුන්දි නොබෝ කලකින් දෙව්ලෝ ගියේය.

ඔහු මළකල ඒ ඇමතියෝ රජකුමරෙක් එනු බලන්නාහු උපතිස්ස ගම වෙසෙමින් රට අනුශාසනා කළෝය. විජහ රජු මළකල රජකුමරුවන් ඊමෙන් පළමුකොට එක් අවුරුද්දක් මේ ලක්දිව අරාජකවි. ඒ සිංහපුයෙහි සිංහබා රජහු ඇවෑමෙන් ඔහු පිත් සුමිත් කුමර රජවි, මිදුරජු දු මෙහෙසියයෙන් ඒ සුමිත් රජු පුතුන් තුන්දෙනෙක් වුහ. ලක්දිවින් ගිය දූතයෝ සිංහපුරයට ගොස් රජහට පත්හසුන් දුන්නෝය. ඒ රජ හසුන් අසා පුතුන් තුන්දෙනාට "දරුවෙනි! මම මහලු යෙමි. තොප අතුරෙන් එකෙක් මා බෑයා සතු නොයෙක් ගුණ ඇති සිත්කලුවු ලඬ්කාවට යේවා. තවද ඔහුගේ ඇවෑමෙන්එහිම රජකරන්ටවත් හොබනේය'ය කීයේය. කිස්දොවුන් මල් පඬුවස්දෙව්යයි නම් රජකුමරතෙමේ යන්නෙමැයි සිතා සැපසේ ගමනද දැන පියහු විසින් අනුදන්නාලද්දේ පරිබ්‍රාමජක වෙස් ඇත්තේ දෙතිස් ඇමැති දරුකෙනෙකුදු ගෙණ නැව් නැංගේය මාකඳුරුගොස හෝ මාඔය මොයෙහි (කම්මල් තොට) ඔව්හු ගොඩ බටහ . ජනතෙමෙ ඒ පරිබ්‍රා◌ිජනකයන් දැක මොනවට සත්කාර කෙළේය.

එ් විජහමිහ රජ ෙත ෙම් අන්තිම වස්හි සිටි ෙ ය් මම ව ෘ ඩ ෙ යමි මා ෙග් පු ෙ තක් න ැ ත් ෙ ත්
ය , දූක ෙ ස් පිතිට්වන ලද   ෙ ම් රට ම ා අ ැ ව ැ ෙ මන් නස් ෙ න්ය , එ ෙ හයින් රජය පිණිස
ම ා මල් සුමිත් කුමරහු ෙ ගන්වන් ෙනමැයි   ෙ ම ෙ ස් සිතූ ෙය්ය . ඉක්බිති අ ැ ම ැ තියන් හ ා
මන්ත්රණය   ෙ ක ා ට   එහි හසුන්පත්යවු ෙ ය්ය . විජහ   ෙ ත ෙ ම් හසුන්දි ෙන ා ෙබා් කලකින්
ෙ දවි ෙලා් ගි ෙ ය්ය. ඔහු මළකල එ් අ ැ මති ෙ ය ා් රජකුම ෙ රක් එනුබලන්න ා හු උපතිස්ස
ගම ෙ ව ෙ සමින් රට අනුශ ා සන ා   ක ෙ ළ ා් ය . විජහරජු මළකල රජකුමර ැ වන් ඊ ෙ මන්
පළමු ෙ ක ා ට   එක් අවුර ැ ද්දක් ෙ ම් ලක්දිව අර ා ජකවි. එ් සි ං හපු ෙ යහි සි ං හබ ා රජහු
අ ැ ව ැ ෙ මන්   ඔහු   පිත් සුමිත් කුමර රජවි , මිදූ රජුදූ ෙ ම ෙ හසිය ෙ යන් එ් සුමිත් රජු
පුතුන් තුන් ෙ ද ෙ නක් වුහ . ලක්දිවින් ගිග් දූන ෙ ය ා් සි ං හපුරයට   ෙ ග ා ස් රජහට පත්හසුන්
දූන් ෙ න ා් ය . එ් රජහසුන් අස ා පුතුන් තුන් ෙ දන ා ට දර ා ෙ වනි මම මහලු ෙ යමි .
ෙ ත ා ප අතු ෙ රන් එ ෙ කක් ම ා බ ැ ය ා සතු ෙ න ා ෙ යක් ගුණඅ ැ ති සිත්කලු වු ල ං ක ා
වට ෙ ය්ව ා . තවද ඔහු ෙ ග් අ ැ ව ැ ෙ මන් එතිම රජකරන්ටවත් ෙ හ ා බ ෙ න්යය කී ෙ ය්ය .
කිස් ෙ ද ා වුන් මල් පඬුවස් ෙ දවියයි නම ී රජකුමර ෙ ත ෙ ම් යන් ෙ නම ැ යි සිත ා ස ැ ප
ෙ ස් ගමනද ද ැ න පියහු විසින් අනුදන්න ා ලද් ෙ ද් පරිබ්ර ා මජක ෙ වස් අ ැ ත් ෙ ත් ෙ දති
ස් අ ැ ම ැ ති දර ා ෙ ක ෙ නකුදූ ෙ ගණ න ැ වින ැ ං ෙ ග්ය ම ා කඳුර ැ ෙ ග ා ස ෙ හා්   ම
ා ඔය ෙ ම ා ෙ යති (කම්මල් ෙ ත ා ට) ඔවිහු ෙ ග ා ඩ බටහ . ජන ෙ ත ෙ ම එ් පරිබ්ර ා ජනක
යන් ද ැ ක ෙම ා නවට සක්ක ා ර ෙක ෙ ළ්ය .

Figure A.3: Test Image 3 (script with "Nirmala" font)


නොව පහළ කළුගමුව හා ඒ අවට පෙදෙස කෙමෙන් පසුපසට තල්ලු වන්නාක් සේ මට දනුණේ ය.

මට සුසන්ත මතක් වෙයි. පුතුන් දෙදෙනාගේ මුහුණු මා ඉදිරියේ පෙනෙයි. මට තෙරුවන් ගුණ සිහි වෙයි. මම පව්කර තිබේ ද යි මෙනෙහි කළෙමි. කුඩා කාලයේ දී දියට වැටුණු කුඹින් ගොඩ ගත් අයුරු සිහි කළෙමි.

මා හදිසියේ පොඩිනා දෙස බැලුවේ ඇය රිට ඔරුව තුළට දමා නැවත හබල අතට ගත් විට ය.

"අපි මෙගොඩට ඇවිත්" .... මට කෑ ගැසිණි.

වෑන්රියේ රියදුරා ද තොටුපළ අසල බලා සිටියි. එතන සිටි ගැමියෙක් මඳක් දියට බැස ඔරුව හරවා ගැනීමට උදව් වූයේ "උඹත් හරි ගෑනි තමා පොඩිනම්මෙ" යි කියමිනි.

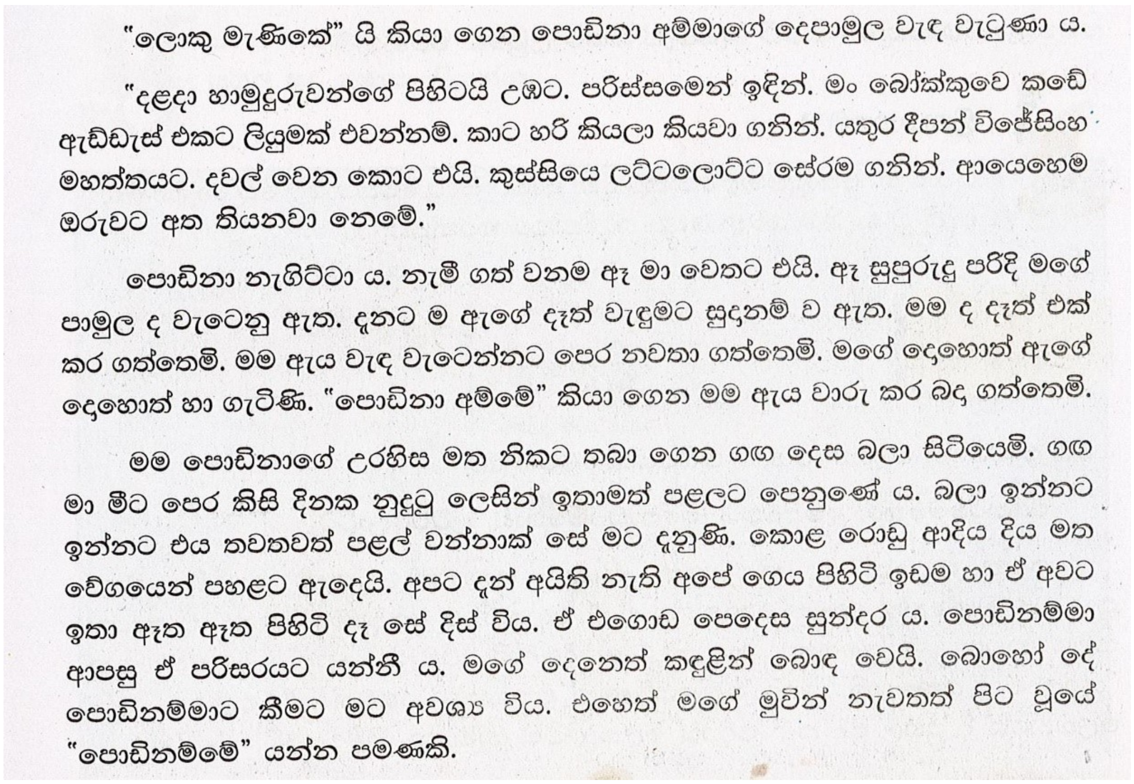
"ඔරුව එනවා දැකලා මාත් බලන් හිටියෙ එගොඩ යන්ඩ. මැණිකෙලාට කතා කරලා වරෙන්. ආපහු යන කොට මං ඔරුව ගෙනියන්නම්".

රියදුරා සූට්කේසයත් ගමන් මලු කීපයකුත් ගෙන වෑන් රිය දෙසට ගියේ ය.

අම්මා කතා නැතිව පොඩිනා දෙස බලා සිටියා ය. මට තවදුරටත් පොඩිනා දුබල මෝඩ ගැහැනියක සේ නො පෙනේ. තමා කරන්නේ කුමක් ද යන්න ගැන ඇයට හොඳ හැඟීමක් ඇත. අප සමග නො ඒමට ඇයට සාධාරණ හේතුවක් ඇතුවාට සැකයක් නැත.

ෙනාව පහළ කළුගමුව හා එ් අවටෙ පෙදසෙ කෙමන් පසුපසට තල්ලුවන්නාක් ෙස් මට දූනුෙණ්ය. මට සුසන්ත මතක් ෙවයි. පුතුන් ෙදෙදනාෙග් මුහුණු මා ඉදිරිෙය් ෙපෙනයි. මටෙ තරැවන් ගුණු සිහිෙවයි. මම පව් කර තිෙබ් දූයි ෙමෙනහිකෙළමි. කුඩා කාලෙය්දී දියට වැටුණු කුඹින් ෙගාඩගත් අයුරැ සිහි කෙළමි. මා හදිසිෙය් ෙපාඩිනා ෙදස බැලුෙව් අ ැය රිට ඔරැව තුළට දමා නැවත හබල අතට ගත්විටය. අපි ෙමෙගාඩට අ ැවිත්............... මට කෑ ගැසිණී. වෑන් රිෙය් රියදුරාද ෙතාටුපළ අසල බලා සිටියි. ා්තන සිටි ගැමිෙයක් මඳක් දියටබැස ඔරැව හරවා ගැනීමට උදව්වූෙය් උඹත් හරි ගැති තමා ෙපාඩිනම්ෙමයි කියමිති. ඔරැව එනවා දූකලා මාත් බලන් හිටිෙය එෙගාඩ යත්ඩ. මැණිෙකලාට කතා කරලා වෙරන්. අ ාපහු යනෙකාට මං ඔරැව ෙගනියන්නම්. රියදූරා සුට්ෙක්සයත් ගමන් මලු කීපයකුත් ෙගන වෑ ැන් රිය ෙදසට ගිෙය්ය. අම්මා කතා නැතිව ෙපංඩිතා ෙදස බලා සිටියාය. මට තවදූරටත් ෙපාඩිනා දුබල ෙමා ්ඩ ගැහැනියක ෙස් ෙනංෙපෙන්. තමා කරත්ෙත් කුමක්ද යත්න ගැත අ ැයට ෙහාඳ හැඟීමක් අ ැත. අප සමග ෙනාඑ ්මට
අ ැයට සාධාරණ ෙහ්තුවක් අ ැතුවාට සැකයක් තැත.

Figure A.4: Test Image 4 (Scanned image) (a) Original Image (b) Output after mapping



"ලොකු මැණිකේ" යි කියා ගෙන පොඩිනා අම්මාගේ දෙපාමුල වැඳ වැටුණා ය.

"දළදා හාමුදුරුවන්ගේ පිහිටයි උඹට. පරිස්සමෙන් ඉඳින්. මං බෝක්කුවෙ කඩේ ඇඩ්ඩැස් එකට ලියුමක් එවන්නම්. කාට හරි කියලා කියවා ගනින්. යතුර දීපන් විජේසිංහ මහත්තයට. දවල් වෙන කොට එයි. කුස්සියෙ ලට්ටලොට්ට සේරම ගනින්. ආයෙහෙම ඔරුවට අත තියනවා නෙමේ."

පොඩිනා නැගිට්ටා ය. නැමී ගත් වනම ඈ මා වෙතට එයි. ඈ සුපුරුදු පරිදි මගේ පාමුල ද වැටෙනු ඇත. දූනට ම ඇගේ දෑත් වැඳුමට සුදානම් ව ඇත. මම ද දෑත් එක් කර ගත්තෙමි. මම ඇය වැඳ වැටෙන්නට පෙර නවතා ගත්තෙමි. මගේ දෙහොත් ඇගේ දෙහොත් හා ගැටිණි. "පොඩිනා අම්මේ" කියා ගෙන මම ඇය වාරු කර බද ගත්තෙමි.

මම පොඩිනාගේ උරහිස මත නිකට තබා ගෙන ගඟ දෙස බලා සිටියෙමි. ගඟ මා මීට පෙර කිසි දිනක නුදුටු ලෙසින් ඉතාමත් පළලට පෙනුණේ ය. බලා ඉන්නට ඉන්නට එය තවතවත් පළල් වන්නාක් සේ මට දැනුණි. කොළ රොඩු ආදිය දිය මත වේගයෙන් පහළට ඇදෙයි. අපට දැන් අයිති නැති අපේ ගෙය පිහිටි ඉඩම හා ඒ අවට ඉතා ඈත ඈත පිහිටි දෑ සේ දිස් විය. ඒ එගොඩ පෙදෙස සුන්දර ය. පොඩිනම්මා ආපසු ඒ පරිසරයට යන්නී ය. මගේ දෙනෙත් කඳුළින් බොඳ වෙයි. බොහෝ දේ පොඩිනම්මාට කීමට මට අවශ්‍ය විය. එහෙත් මගේ මුවින් නැවතත් පිට වූයේ "පොඩිනම්මේ" යන්න පමණකි.

ෙ ල ා කු ම ැ ණි ශ ක් යි කි ය ා ෙ ග න ෙ ප ා ඩි න ා අ ම් ම ා ෙ ග් ශ ද ප ා මු ල ව ැ ඳ ව ැ ටු ණු ා ය .
ද ළ ද ා හ ා මු දු ර ැ ව න් ෙ ග් පි හි ට යි උ ඹ ට . ප රි ස් ස ෙ ම ත් ඉ දි ත් . ම ං ෙ බ ා ් ක් කු ෙ ව ක ෙ ඩ
අ ැ ඩ ඩ ැ ස් එ ක ට ලි යු ම ක් එ ව න් න ම් . ක ා ට හ රි කි ය ල ා කි ය ව ා ග ති ත් . ය තු ර දී ප න් වි ෙ ජ් සි ං හ
ම හ ත් ත ය ට . ද ව ල් ම ව න ශ ක ා ට එ යි . කු ස් සි ශ ය ල ට් ට ෙ ල ා ට් ට ශ ස් ර ම ග ති ත් . අ ා ෙ ය ෙ හ ම
ඔ ර ැ ව ට අ ත ති ය න ව ා ශ න ශ ම් . ෙ ප ා ඩි න ා න ැ ගි ට් ට ා ය . න ැ මී ග ත් ව න ම අ ැ ම ා
ශ ව ත ට එ යි . අ ැ සු පු ර ැ දු ප රි දි ම ශ ග් ප ා මු ල ද ව ැ ශ ට නු අ ැ ත . ද න ට ම අ ැ ෙ ග් ද ැ ත්
ව ැ ඳු ම ට සු ද ා න ම් ව අ ැ ත . ම ම ද දැ ත් එ ක් ක ර ග ත් ශ ත ම් . ම ම අ ැ ය ව ැ ඳ ව ැ ශ ට න් න ට
ෙ ප ර න ව ත ා ග ත් ශ ත ම් . ම ෙ ග් ශ ද ා ශ හ ා ත් අ ැ ශ ග් ෙ ද ා ෙ හ ා ත් හ ා ග ැ ටි ණි .
ෙ ප ා ඩි න ා අ ම් ෙ ම් කි ය ා ශ ග න ම ම අ ැ ය ව ා ර ැ ක ර බ ද ා ග ත් ශ ත ම් . ම ම
ෙ ප ා ඩි න ා ශ ග් උ ර හි ස ම ත නි ක ට තු බ ා ශ ග න ග ඟ ෙ ද ස බ ල ා සි ටි ශ ය ම් . ග ඟ ම ා මී ට
ශ ප ර කි සි දි න ක නු දු වු ශ ල සි න් ඉ ත ා ම ත් ප ළ ල ට ශ ප නු ශ ණ් ය . බ ල ා ඉ ත් න ට ඉ න් න ට එ ය ත ව
ත ව ත් ප ළ ල් ව න් න ා ක් ශ ස් ම ට ද නු ණි . ෙ ක ා ළ ම ර ා ඩු අ ා දි ය දි ය ම ත ශ වි ග ෙ ය න් ප හ ළ ට
අ ැ ම ද යි . අ ප ට ද න් අ යි ති න ැ ති අ ෙ ප් ශ ග ය ෂි හි ටි ඉ ඩ ම හ ා එ ් අ ව ට ඉ ත ා අ ැ ත අ ැ ත
ෂි හි ටි ද ැ ශ ස් දි ස් වි ය . එ ් එ ශ ග ා ඩ ශ ප ශ ද ස සු ත් ද ර ය . ෙ ප ා ඩි න ම් ම ා අ ා ප සු එ ් ප රි ස ර ය ට
ය ත් නී ය . ම ෙ ග් ෙ ද ශ න ත් ක ඳු ළි ත් ම බ ා ඳ ශ ව යි . ෙ බ ා ග හා ් ෙ ද් ෙ ප ා ඩි න ම් ම ා ට කී ම ට
ම ට අ ව ශ ා ය වි ය . එ ෙ හ ත් ම ශ ග් මු වි ත් න ැ ව ත ත් පි ට වූ ශ ය් ෙ ප ා ඩි න ම් ෙ ම් ය න් න ප ම ණ කි

Figure A.5: Test Image 5 (Scanned image) (a) Original Image (b) Output after mapping

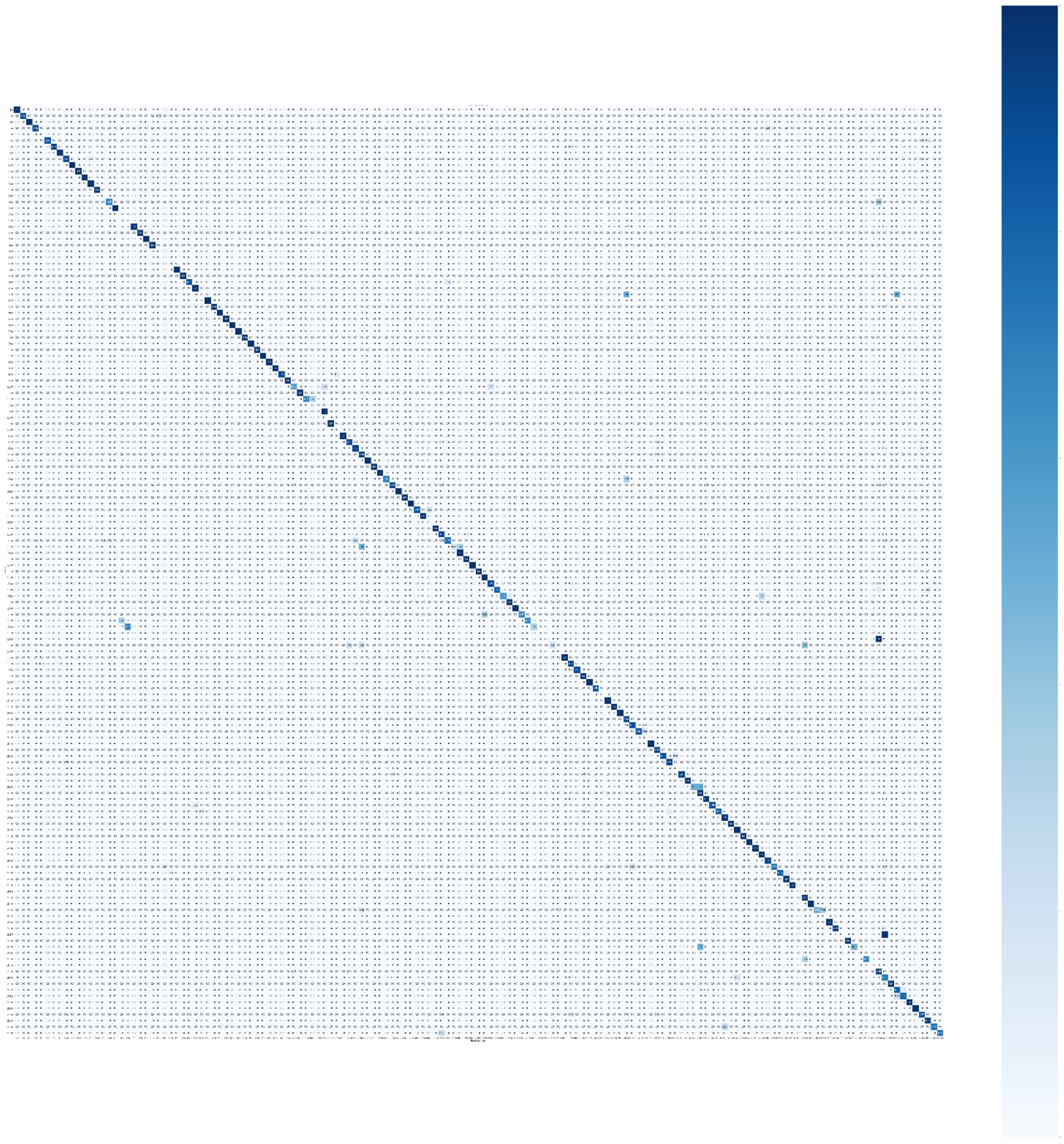
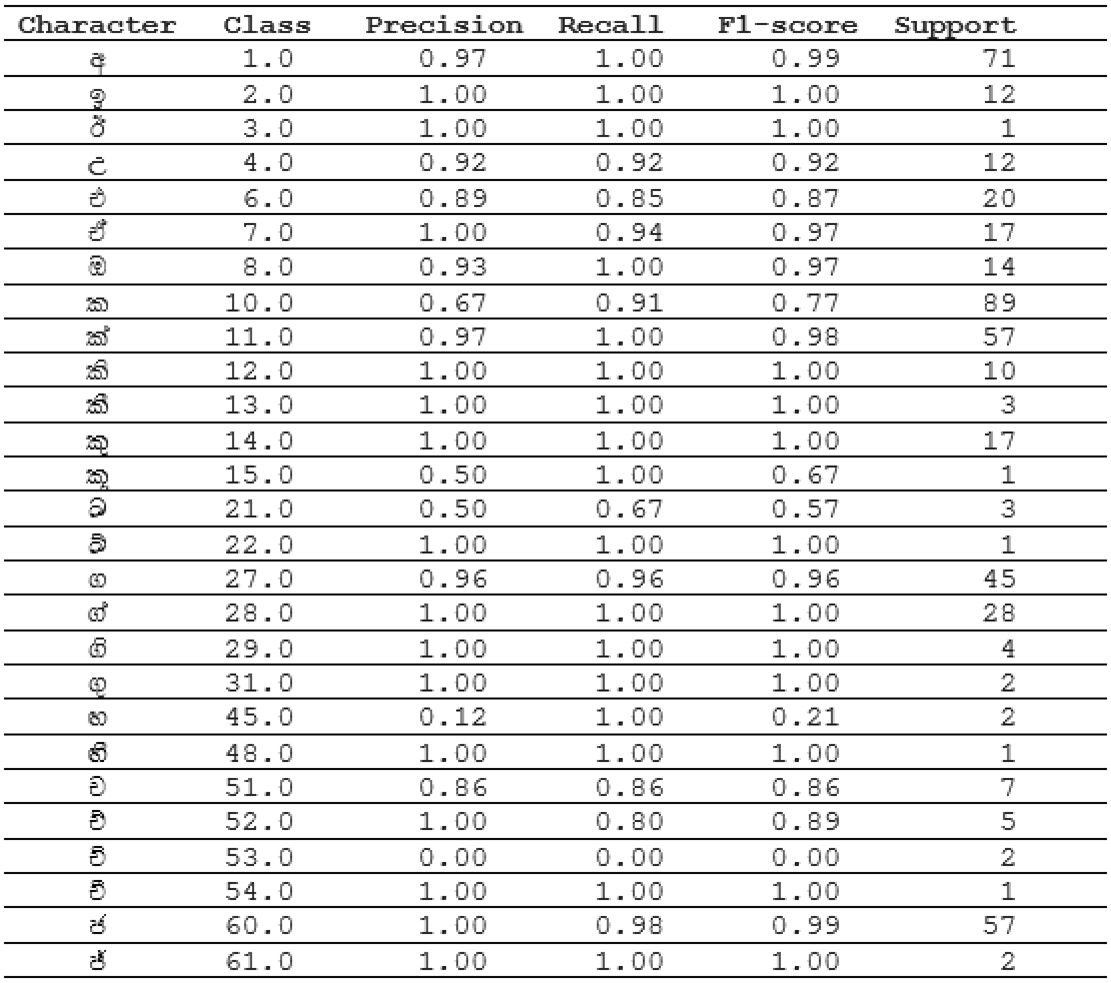
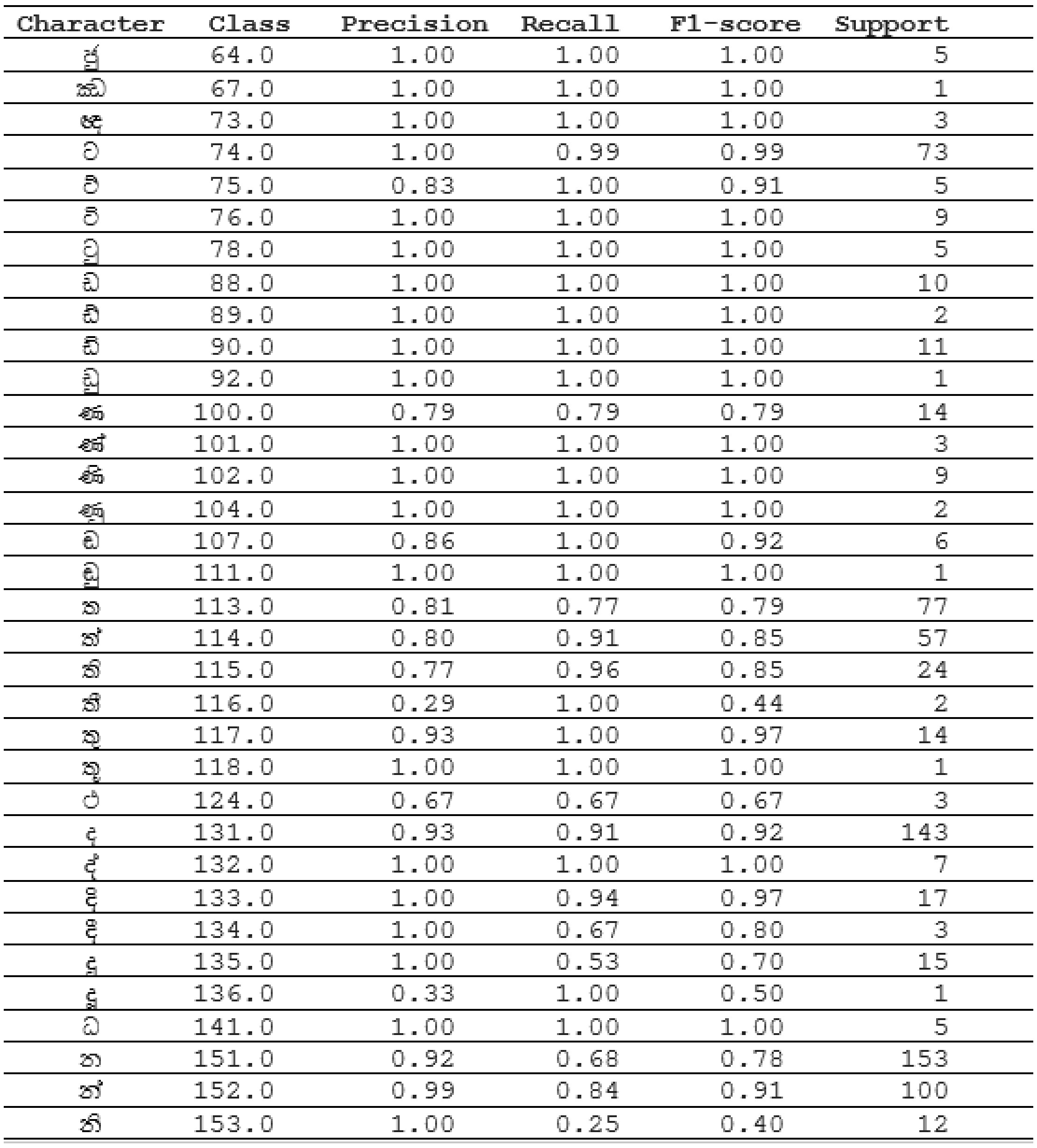


Figure A.6: Confusion matrix of the proposed classifier



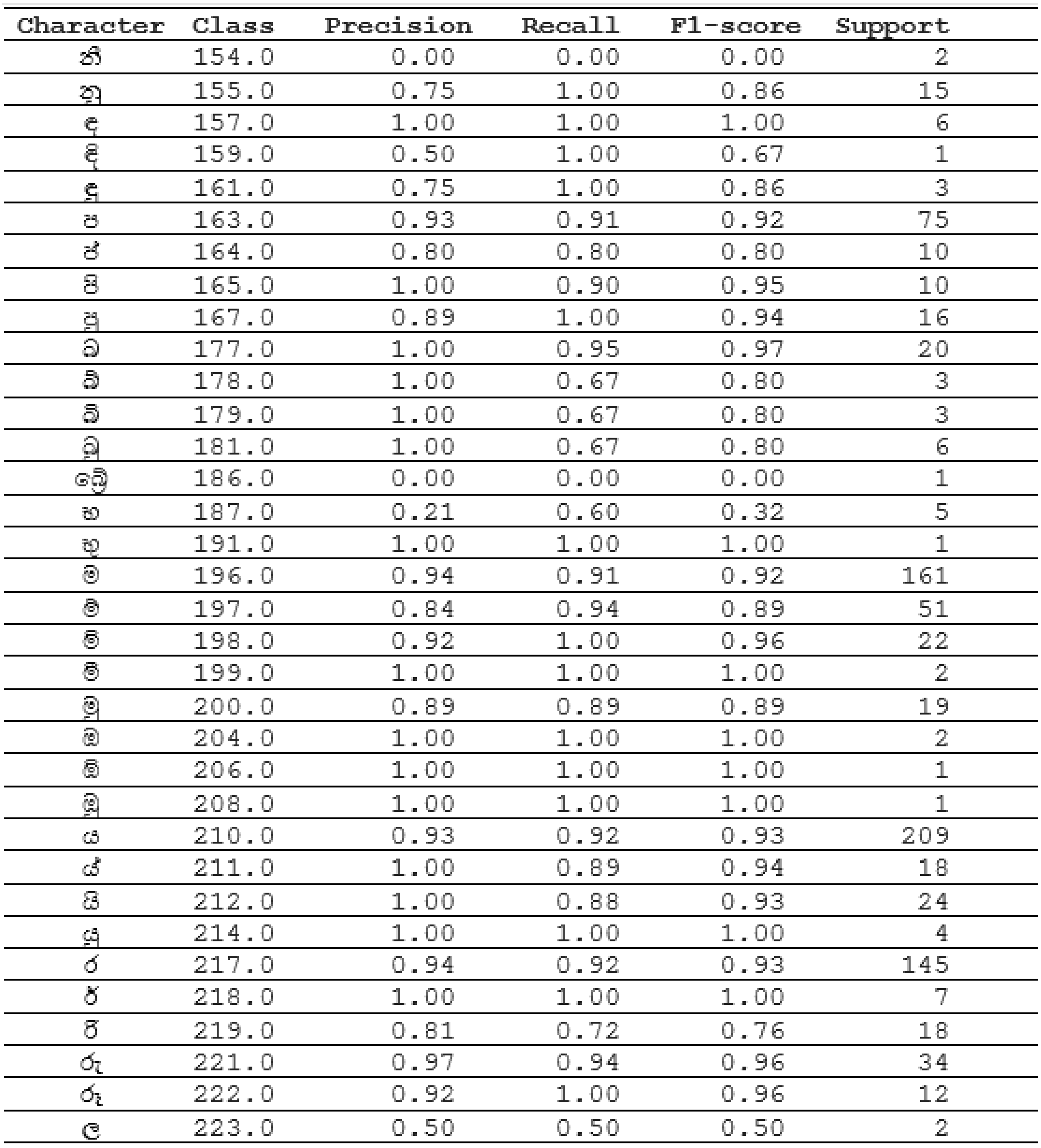
Character	Class	Precision	Recall	F1-score	Support
අ	1.0	0.97	1.00	0.99	71
ඉ	2.0	1.00	1.00	1.00	12
ඊ	3.0	1.00	1.00	1.00	1
උ	4.0	0.92	0.92	0.92	12
එ	6.0	0.89	0.85	0.87	20
ඒ	7.0	1.00	0.94	0.97	17
ඔ	8.0	0.93	1.00	0.97	14
ක	10.0	0.67	0.91	0.77	89
ක්	11.0	0.97	1.00	0.98	57
කි	12.0	1.00	1.00	1.00	10
කී	13.0	1.00	1.00	1.00	3
කු	14.0	1.00	1.00	1.00	17
කූ	15.0	0.50	1.00	0.67	1
ට	21.0	0.50	0.67	0.57	3
ට්	22.0	1.00	1.00	1.00	1
ග	27.0	0.96	0.96	0.96	45
ග්	28.0	1.00	1.00	1.00	28
ගි	29.0	1.00	1.00	1.00	4
ගු	31.0	1.00	1.00	1.00	2
ත	45.0	0.12	1.00	0.21	2
තී	48.0	1.00	1.00	1.00	1
ව	51.0	0.86	0.86	0.86	7
ව්	52.0	1.00	0.80	0.89	5
වි	53.0	0.00	0.00	0.00	2
වී	54.0	1.00	1.00	1.00	1
ප	60.0	1.00	0.98	0.99	57
ප්	61.0	1.00	1.00	1.00	2

Figure A.7: Precision, recall, f1-score for approach 1



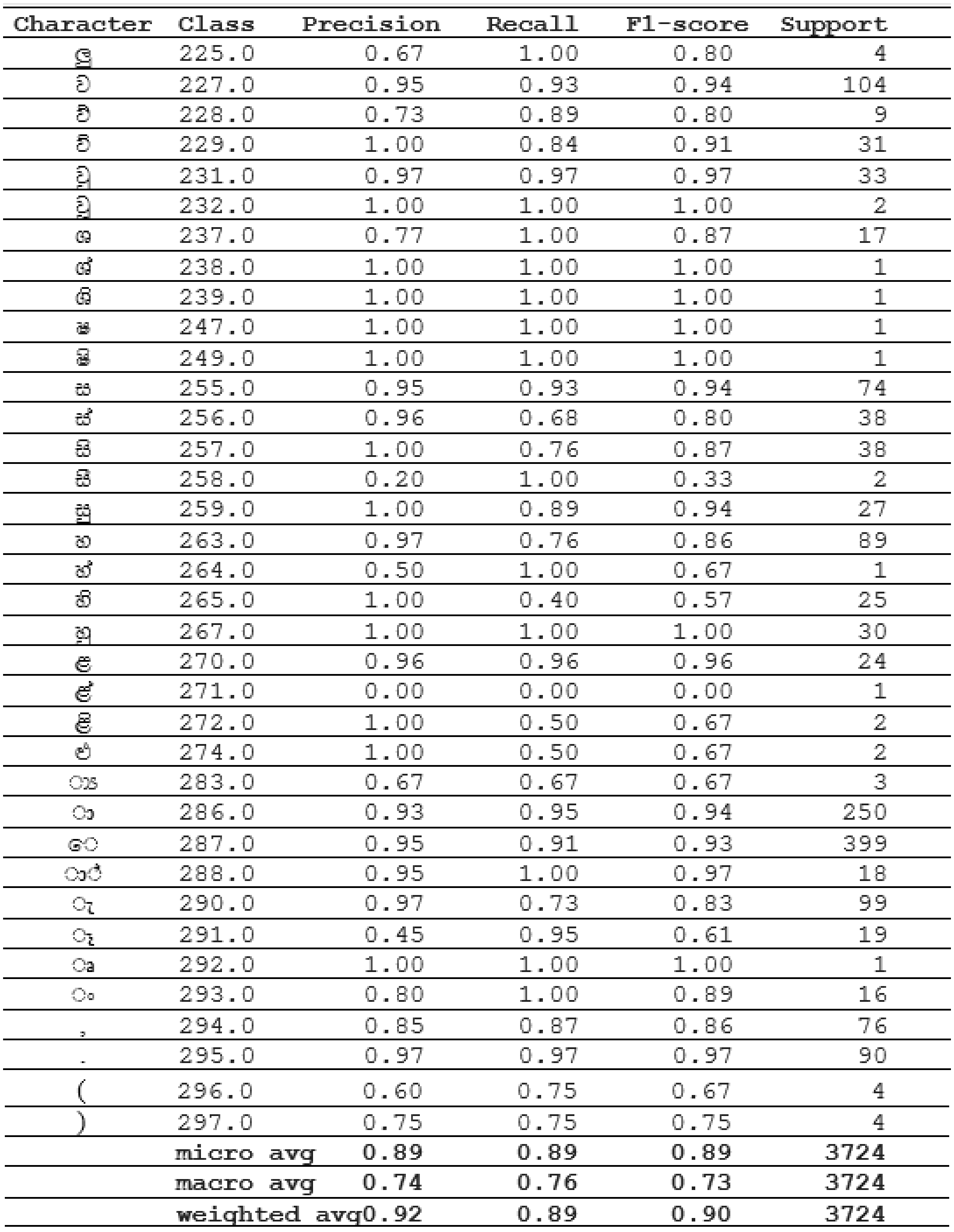
Character	Class	Precision	Recall	F1-score	Support
ජු	64.0	1.00	1.00	1.00	5
ඣ	67.0	1.00	1.00	1.00	1
ඤ	73.0	1.00	1.00	1.00	3
ට	74.0	1.00	0.99	0.99	73
ටි	75.0	0.83	1.00	0.91	5
ටී	76.0	1.00	1.00	1.00	9
ටු	78.0	1.00	1.00	1.00	5
ඩ	88.0	1.00	1.00	1.00	10
ඩි	89.0	1.00	1.00	1.00	2
ඩී	90.0	1.00	1.00	1.00	11
ඩු	92.0	1.00	1.00	1.00	1
ණ	100.0	0.79	0.79	0.79	14
ණ්	101.0	1.00	1.00	1.00	3
ණි	102.0	1.00	1.00	1.00	9
ණු	104.0	1.00	1.00	1.00	2
ඬ	107.0	0.86	1.00	0.92	6
ඬු	111.0	1.00	1.00	1.00	1
ත	113.0	0.81	0.77	0.79	77
ත්	114.0	0.80	0.91	0.85	57
ති	115.0	0.77	0.96	0.85	24
තී	116.0	0.29	1.00	0.44	2
තු	117.0	0.93	1.00	0.97	14
තූ	118.0	1.00	1.00	1.00	1
ථ	124.0	0.67	0.67	0.67	3
ද	131.0	0.93	0.91	0.92	143
ද්	132.0	1.00	1.00	1.00	7
දි	133.0	1.00	0.94	0.97	17
දී	134.0	1.00	0.67	0.80	3
දු	135.0	1.00	0.53	0.70	15
දූ	136.0	0.33	1.00	0.50	1
ධ	141.0	1.00	1.00	1.00	5
න	151.0	0.92	0.68	0.78	153
න්	152.0	0.99	0.84	0.91	100
නි	153.0	1.00	0.25	0.40	12

Figure A.8: Precision, recall, f1-score for approach 1



Character	Class	Precision	Recall	F1-score	Support
නී	154.0	0.00	0.00	0.00	2
නු	155.0	0.75	1.00	0.86	15
ඳ	157.0	1.00	1.00	1.00	6
ඳි	159.0	0.50	1.00	0.67	1
ඳු	161.0	0.75	1.00	0.86	3
ප	163.0	0.93	0.91	0.92	75
පි	164.0	0.80	0.80	0.80	10
පී	165.0	1.00	0.90	0.95	10
පු	167.0	0.89	1.00	0.94	16
බ	177.0	1.00	0.95	0.97	20
බි	178.0	1.00	0.67	0.80	3
බී	179.0	1.00	0.67	0.80	3
බු	181.0	1.00	0.67	0.80	6
බෞ	186.0	0.00	0.00	0.00	1
භ	187.0	0.21	0.60	0.32	5
භු	191.0	1.00	1.00	1.00	1
ම	196.0	0.94	0.91	0.92	161
මි	197.0	0.84	0.94	0.89	51
මී	198.0	0.92	1.00	0.96	22
මී	199.0	1.00	1.00	1.00	2
මු	200.0	0.89	0.89	0.89	19
ඹ	204.0	1.00	1.00	1.00	2
ඹි	206.0	1.00	1.00	1.00	1
ඹු	208.0	1.00	1.00	1.00	1
ය	210.0	0.93	0.92	0.93	209
යි	211.0	1.00	0.89	0.94	18
යී	212.0	1.00	0.88	0.93	24
යු	214.0	1.00	1.00	1.00	4
ර	217.0	0.94	0.92	0.93	145
රි	218.0	1.00	1.00	1.00	7
රී	219.0	0.81	0.72	0.76	18
රු	221.0	0.97	0.94	0.96	34
රූ	222.0	0.92	1.00	0.96	12
ල	223.0	0.50	0.50	0.50	2

Figure A.9: Precision, recall, f1-score for approach 1



Character	Class	Precision	Recall	F1-score	Support
ලු	225.0	0.67	1.00	0.80	4
ව	227.0	0.95	0.93	0.94	104
වි	228.0	0.73	0.89	0.80	9
වී	229.0	1.00	0.84	0.91	31
වු	231.0	0.97	0.97	0.97	33
වූ	232.0	1.00	1.00	1.00	2
ශ	237.0	0.77	1.00	0.87	17
ශ්	238.0	1.00	1.00	1.00	1
ශි	239.0	1.00	1.00	1.00	1
ෂ	247.0	1.00	1.00	1.00	1
ෂි	249.0	1.00	1.00	1.00	1
ස	255.0	0.95	0.93	0.94	74
ස්	256.0	0.96	0.68	0.80	38
සි	257.0	1.00	0.76	0.87	38
සී	258.0	0.20	1.00	0.33	2
සු	259.0	1.00	0.89	0.94	27
හ	263.0	0.97	0.76	0.86	89
හ්	264.0	0.50	1.00	0.67	1
හි	265.0	1.00	0.40	0.57	25
හු	267.0	1.00	1.00	1.00	30
ළ	270.0	0.96	0.96	0.96	24
ළ්	271.0	0.00	0.00	0.00	1
ළු	272.0	1.00	0.50	0.67	2
ෆ	274.0	1.00	0.50	0.67	2
්‍ය	283.0	0.67	0.67	0.67	3
ා	286.0	0.93	0.95	0.94	250
ෙ	287.0	0.95	0.91	0.93	399
ෟ	288.0	0.95	1.00	0.97	18
ැ	290.0	0.97	0.73	0.83	99
ෑ	291.0	0.45	0.95	0.61	19
ෘ	292.0	1.00	1.00	1.00	1
ං	293.0	0.80	1.00	0.89	16
,	294.0	0.85	0.87	0.86	76
.	295.0	0.97	0.97	0.97	90
(	296.0	0.60	0.75	0.67	4
)	297.0	0.75	0.75	0.75	4
	micro avg	**0.89**	**0.89**	**0.89**	**3724**
	macro avg	**0.74**	**0.76**	**0.73**	**3724**
	weiqhted avg	**0.92**	**0.89**	**0.90**	**3724**

Figure A.10: Precision, recall, f1-score for approach 1