LONTAR KOMPUTER VOL. 13, NO. 3 DECEMBER 2022 **p-ISSN 2088-1541**
DOI : 10.24843/LKJITI.2022.v13.i03.p03 **e-ISSN 2541-5832**
Accredited Sinta 2 by RISTEKDIKTI Decree No. 158/E/KPT/2021

Balinese Script Recognition Using Tesseract Mobile Framework

Gede Indrawan[a1], Ahmad Asroni[a2], Luh Joni Erawati Dewi[a3], I Gede Aris Gunadi[a4], I Ketut Paramarta[b5]

[a]Department of Electrical Engineering and Computer Science, Universitas Pendidikan Ganesha
Jl. Udayana 11, Singaraja, Buleleng, Bali, Indonesia
[1]gindrawan@undiksha.ac.id (Corresponding author)
[2]ahmad.asroni@undiksha.ac.id
[3]joni.ernawati@undiksha.ac.id
[4]igedearisgunadi@undiksha.ac.id

[b]Department of Balinese Language Education, Universitas Pendidikan Ganesha
Jl. Ahmad Yani 67, Singaraja, Buleleng, Bali, Indonesia
[5]ketut.paramarta@undiksha.ac.id

Abstract

One of the main factors causing the decline in the use of Balinese Script is that Balinese people are less interested in reading Balinese Script because of their reluctance to learn Balinese Script, which is relatively complicated in the recognition process. The development of computer technology has now been used to help by performing character recognition or known as Optical Character Recognition (OCR). Developing the OCR application for Balinese Script is an effort to help preserve, from the technology side, as a means of education related to Balinese Script. In this study, that development was conducted by using a Tesseract OCR engine that consists of several stages, i.e., the first one is to prepare the dataset, the second one is to generate the dataset using the Web Scraping method, the third one is to train the OCR engine using the generated dataset, and finally, the fourth one is to implement the generated language model into a mobile-based application. The study results prove that the dataset generation process using the Web Scraping method can be a better choice when faced with a training dataset that requires a large dataset compared to several previous studies of non-Latin character recognition. In those studies, the jTessBox tools were used, which took time because they had to select per character for a dataset. The best result of the language model is a combination of character, word, sentence, and paragraph datasets (hierarchical combination of character, word, sentence, and paragraph datasets) with a coincidence rate of 66.67%. The more diverse and structured hierarchical datasets used, the higher the coincidence rate.

Keywords: *Balinese Script, Mobile Framework, Tesseract, Optical Character Recognition, Web Scraping*

1. Introduction

Balinese Script, literature, and language are sources of imagination, creativity, and energy in Balinese culture. This is starting to decline, especially in terms of the use of Balinese Script, which is decreasingly being used in the daily life of Balinese people [1]. One of the main factors causing the decline is that Balinese people are less interested in reading Balinese Script because of their reluctance to learn Balinese Script, which is relatively complicated in the recognition process. Bali Governor Regulation Number 80 of 2018, concerning the protection and use of the Balinese Language, Script, and literature, also the Implementation of the Balinese Language Month, regulates the use of the Balinese language as a means of communication in Balinese family life, communication in all activities of Hindu religious, Balinese customs and culture, and providing information on public services both in government institutions and private institutions as a companion to Indonesian [2].

The development of computer technology has now been widely used to perform character recognition, termed Optical Character Recognition (OCR). OCR converts printed text and images

into digital character forms, which machines can manipulate. OCR implementation has been used in many application sectors, such as education, banking, finance, law, etc. Along with the development of OCR technology, many studies have used OCR to perform character recognition for non-Latin scripts [3]. Most of the development of OCR is still focused on Latin English script because it is supported by the encoding standard of the American Standard Code for Information Interchange or ASCII for short. The limited ability of OCR to recognize non-Latin scripts is a challenge for researchers to improvise.

OCR technology is growing rapidly with the creation of several OCR engines that are open source and paid. This study tested which OCR engine has the highest performance for Information Extraction using Named Entity Recognition by comparing three OCR engines, namely Foxit, PDF2GO, and Tesseract [4]. Based on the research conducted by Ramdhani et al., compared the performance levels of three OCR engines with high-performance levels. The test was carried out with 8,562 government human resource documents in six document categories, two document structures, and four measurements. The test results found that Tesseract was the most suitable solution and got the highest performance in Information Extraction. The details of the test results, on average, PDF2GO gets a performance of 86.27%, Foxit gets a performance value of 84.01%, and Tesseract gets a performance value of 92.46%.

In a study by Abdul Robby et al., they used the Tesseract OCR engine to be implemented as a Javanese Script character recognition engine. This study aims to simplify the process of automatically recognizing Javanese characters using a mobile application [5]. The dataset used as a data source to build the Tesseract OCR engine training data is 5,880 Javanese characters. To build the Javanese Script dataset was collected from digital characters with specifications (3 sets x 120 characters) and handwriting (46 sets x 120 characters). The dataset training tools used in this study are the Neural-Network API from the Tesseract OCR engine. Before the training, the Javanese Script dataset was selected by segmenting each character and setting variables for the cluster of characters using JTessBoxEditor. The highest accuracy achieved by the model generated from the trained data is 97.50%.

The following research similar to the case of non-Latin optical character recognition is the study conducted by Mudiarta et al. This research focuses on preserving knowledge of reading Balinese Script in pictures by combining information technology with Balinese Script discipline. In this study, the OCR application was developed on a mobile-based device with camera facilities. The input in this application is in the form of images and is processed with Tesseract OCR engine technology. The Balinese Script dataset is based on eighteen basic Balinese Script syllables and only numbers to carry out the training process. The tool used to carry out the training process is jTessBoxEditor. This tool has fully automated facilities for training datasets. In the test results for 50 words, 62% recognition was obtained with good quality image-based Bali-Simbar font [1].

From the exposure of the two studies above, there are similarities in terms of the Optical Character Recognition engine and the data training process carried out. The training data to create the trained data model utilizes the jTessBoxEditor tool by segmenting characters from non-Latin character images. The segmentation process is carried out alternately for each dataset owned. The jTessBoxEditor tools must be done manually by segmenting each dataset, making the training process relatively more time-consuming. Several weaknesses occur in the two studies, especially in the data training process. In the chapter suggestions of the two studies, the focus is on increasing the number of datasets used.

Based on the weaknesses and suggestions of the two studies, it can be resolved using different data training methods. In addition to using the jTessBoxEditor tools, there is the latest training method to create trained data, using the latest Tesseract OCR training method. The latest version of Tesseract OCR provides training tools without relying on external tools such as jTessBoxEditor. The concept of training datasets in the newest version of Tesseract OCR tools supports the automatic dataset training process by using the command line for all dataset training execution commands. Compared to jTessBoxEditor, almost all steps must be done manually using a GUI, such as selecting the character box segmentation, correcting the ground truth character box, and merging all the resulting training data files. This latest Tesseract OCR training method can perform dataset training simultaneously for all datasets. According to Idrees & Hassani, since version 4.0, Tesseract OCR presents a new engine based on Long Short-Term Memory (LSTM) [6]. LSTM, as a special form of Artificial Neural Network (RNN), provides much higher accuracy

in image recognition than the previous version of Tesseract OCR. In the previous version, Tesseract OCR processing still used traditional processing step by step, not using artificial neural network (RNN). In the first stage of connected component analysis, the outline is collected and will be converted into a Blob.

Furthermore, in the second stage, Blob will be arranged into proportional text lines, broken down into words with definite and fuzzy spaces. The third stage is character recognition, namely the recognition of each word, and the last is validating alternative hypotheses to find lowercase text using fuzzy space [7]. The Tesseract can be trained from scratch or refined based on the language that has been trained.

2. Research Methods

This research focuses on applying the latest Tesseract OCR training model for non-Latin digital characters, especially languages that Tesseract OCR has not supported. No research has been found regarding this. This study uses the latest data training method from Tesseract OCR by focusing on the dataset format consisting of two types of datasets, namely the image and the ground truth image. This training method differs from the two studies that discuss non-Latin digital character recognition using the jTessBoxEditor tools to conduct data training [1][5]. The stages carried out in this study can be seen in Figure 1.

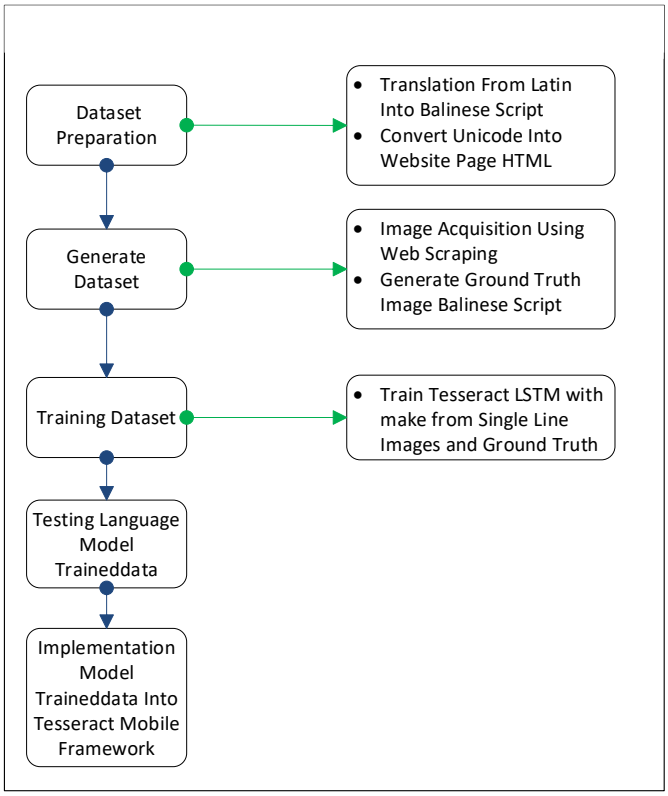


Figure 1. Research Methodology

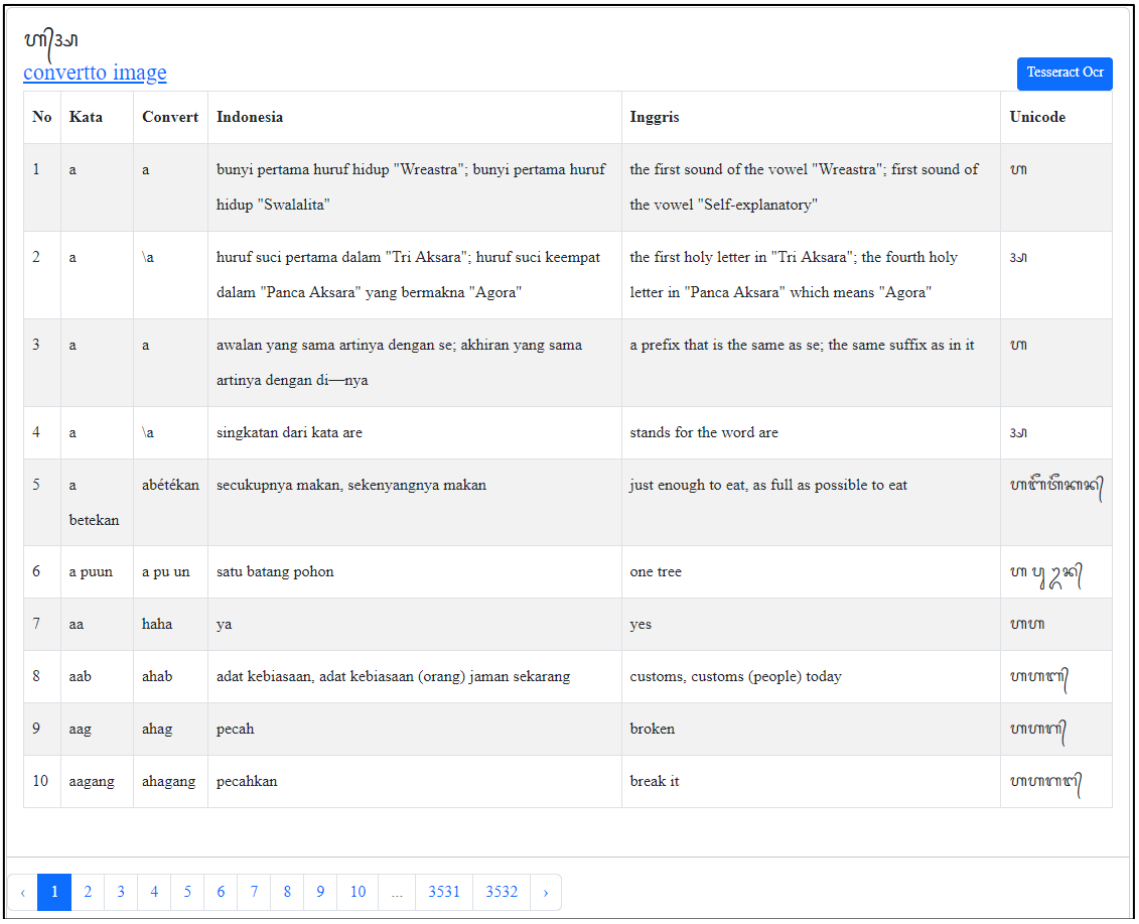
2.1. Dataset Preparation

Dataset preparation was carried out to obtain a data set consisting of character images and ground truth. The data used to create the dataset is derived from the research conducted by G. Indrawan et al. [8]. That research consolidated a dataset with more than 35,000 words in Balinese with its Indonesian and English counterparts. The transliteration method implemented in the study was adopted using a different platform, namely using a website-based platform. The preparation of this dataset went through several stages for transliteration from Latin to Balinese Script. The first stage was converting the Latin-Balinese dataset into the database using Unicode to display

on HTML pages. Next, add the family Balinese font Noto Sans Balinese so that the Unicode displayed on the HTML page can be converted into digital Balinese characters. The results of the dataset preparation can be seen in Figure 2.

2.2. Dataset Generation

The dataset generation technique used to extract information from the website platform is the web scraping technique [9]. The web scraping technique extracts information from websites automatically by parsing hypertext tags and retrieving information in the form of text, images, and videos embedded in them from large amounts of data from web pages [10][11]. The web scraping technique implemented in this research consists of four main processes. The first process is to create a scrapping template in the form of an HTML page that contains information that can be extracted into a Balinese Script image dataset, and the Balinese Script ground truth. The second process runs the website using the browser in the browser search field. The third process is making a web scraping algorithm to acquire Balinese Script images and automatically extracting ground truth when the algorithm is run. The last process is to store all the datasets resulting from the web scraping technique in the database. The dataset generated from the web scraping process consists of two datasets: the Balinese Script image dataset and the Balinese Script ground truth in digital character format.


convertto image

Tesseract Ocr

No	Kata	Convert	Indonesia	Inggris	Unicode
1	a	a	bunyi pertama huruf hidup "Wreastra"; bunyi pertama huruf hidup "Swalalita"	the first sound of the vowel "Wreastra"; first sound of the vowel "Self-explanatory"	ᬅ
2	a	\a	huruf suci pertama dalam "Tri Aksara"; huruf suci keempat dalam "Panca Aksara" yang bermakna "Agora"	the first holy letter in "Tri Aksara"; the fourth holy letter in "Panca Aksara" which means "Agora"	ᬅ
3	a	a	awalan yang sama artinya dengan se; akhiran yang sama artinya dengan di—nya	a prefix that is the same as se; the same suffix as in it	ᬅ
4	a	\a	singkatan dari kata are	stands for the word are	ᬅ
5	a betekan	abétékan	secukupnya makan, sekenyangnya makan	just enough to eat, as full as possible to eat	ᬅᬩᬾᬢᬾᬓᬦ᭄
6	a puun	a pu un	satu batang pohon	one tree	ᬅ ᬧᬸ ᬉᬸᬦ᭄
7	aa	haha	ya	yes	ᬳᬳ
8	aab	ahab	adat kebiasaan, adat kebiasaan (orang) jaman sekarang	customs, customs (people) today	ᬳᬳᬩᬶ
9	aag	ahag	pecah	broken	ᬳᬳᬕᬶ
10	aagang	ahagang	pecahkan	break it	ᬳᬳᬕᬗᬶ

‹ 1 2 3 4 5 6 7 8 9 10 ... 3531 3532 ›

Figure 2. Result of Dataset Preparation

2.3. Dataset Training

As a well-known open-source OCR engine, Tesseract [12] is under active development by Google. It is currently available with the latest version 5.0, including the newest version of the LSTM-based OCR engine. Meanwhile, other Tesseract version below 5.0 is categorized as traditional machine [13]. LSTM is a Recurrent Neural Network in Deep Learning developed

specifically for handling sequential prediction problems [14]. Tesseract can be trained using several operating systems, such as Linux, Windows, and macOS, by running a command line set and the Tesseract OCR training shell script [15]. Several operating system options can be selected according to needs, but Tesseract OCR is recommended to use the Linux operating system locally or in the cloud. A virtual server has a relatively good performance in running data training. In their use, containers have various benefits or advantages that make them popular among data training tools, such as having a simple configuration, good security level, can run on several cloud platforms, can perform debugging, and can be used on various operating systems [16]. The dataset training consists of two main processes: character form training and language dictionary creation. The output of the dataset training is the trained data file that needs to be copied to the Tesseract instance data folder and will be used to perform character recognition.

2.4. Language Model Testing

Testing the language model is an important stage to test the language model generated from the dataset training process. The result of trained data obtained after training the dataset through a testing process consisting of two types of testing, namely the unit testing and performance testing stages [17][18]. To perform automated unit testing, some additional requirements are required. It includes additional dependencies for training tools and downloads all necessary submodules, such as git and the model repository. In comparison, performance testing is carried out to obtain test results to see the model's level of speed and performance based on the allocation of resources used [15]. One of the unit testing methods that can be used to measure the language model's accuracy is Coincidence. Coincidence refers to the accuracy level of an optical character recognition language model. The way Coincidence work is to do a match based on an identifiable character matrix. The matrix form in question is a single-line transliteration to the ground truth of the testing character image. The accuracy test result using the Coincidence method is the percentage level of accuracy. A higher level of Coincidence means that the accuracy of the language model is also higher. Still, if the level of Coincidence is low, it means that the quality of the accuracy of the language model is also low [19][20]. A step that can be taken to optimize the model's performance is to optimize the code to increase memory capability in processing large numbers of characters. Much better performance improvements can be made by making the network smaller [21].

2.5. Tesseract Mobile Framework

The mobile framework technology used in this research is the Flutter Mobile Framework. Flutter is an open-source UI kit developed by Google that allows the creation of cross-platform applications, including Android and IOS platforms. Flutter was first introduced at the 2015 Dart Developer Summit. On December 4th, 2018, Google released Flutter 1.0 at the Flutter Live Event. This also marks the release of the first stable version of Flutter. Subsequently, Flutter 1.12 was released at the Flutter Interact event on December 11, 2019 [22]. Flutter supports cross-platform that can be run on several different platforms. By using Flutter, the Android and iOS application development process can be done at the same time.

Other than mobile platforms, Flutter can also run on web and desktop platforms. This will save time by not needing to learn the native language used on each platform. As a result, developers can produce high-quality applications that run well on multiple platforms using only one codebase [23]. Flutter uses Dart programming language, which Google also created in 2011. The Flutter engine is mainly written in the C++ programming language and remains at the core of Flutter. The engine implements Flutter's core APIs, including accessibility support, Dart runtime, text graphics layout, and plugin architecture. Flutter consists of a system layer structure. It works and runs in order, with each layer depending on the previous layer [24]. With the advantage offered by Flutter in the development process, namely one codebase for multi-platforms, it can provide a level of code efficiency that can be increased. In principle, the flutter system development applies the concept of reusable widgets, where the basic architecture of Flutter can be seen in Figure 3.

p-ISSN 2088-1541
e-ISSN 2541-5832

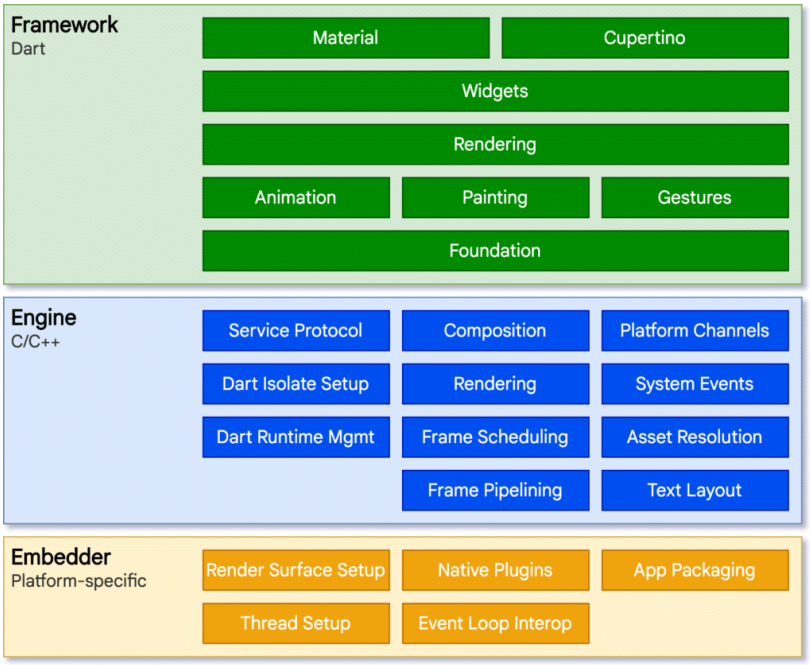


Figure 3. Flutter Basic Architecture

3. Result and Discussion

Balinese Script Optical Character Recognition uses Tesseract OCR engine version 5 as the model and Flutter mobile framework version 2.16 as the mobile application framework. In the dataset training stage, the operating system used is a Linux Ubuntu 20.04 virtual server with specifications of 1 GB Memory, 25 GB Disk, and SGP1 - Ubuntu 20.04 (LTS) x64. For the dataset training process to run in an isolated environment, a service is needed that provides the ability to package and run an application in an isolated environment called a container. With adequate isolation and security, running multiple containers simultaneously on a particular host is possible. In this section, the discussion related to the research results consists of several sections based on the two main technologies used: Tesseract OCR and Flutter Mobile Framework.

3.1. Dataset Generation Result

Generating datasets using the web scrapping method aims to produce two datasets: the Balinese Script image dataset and ground truth transliteration. The process of generating data requires Balinese language data, which is converted into Balinese Script using Unicode. The Balinese language data used is a Balinese transliteration dataset totaling 35,319 words. The composition of the transliterated dataset consists of Balinese, Indonesian and English words. The amount of data based on the word index of the dataset can be seen in Table 1.

Based on the composition of the transliterated data in Table 1, it is then converted into a pair of data, namely a single-line text image with a "png" file extension and its single-line transliteration text with a "gt.txt" file extension. The form of the resulting dataset can consist of text images of the alphabet and text images of words in Balinese. At the dataset generation stage, a website-based platform uses the Laravel framework as a backend. In addition to using the backend at this stage, the other plugin for the image acquisition process that works on the client side was used. This plugin aims to ease server performance in generating a large number of datasets. This image acquisition process captures selected HTML pages based on the index id of each element simultaneously. Using an id on each HTML element aims to provide a unique identity so that when the image acquisition plugin performs image capture, it can select the area's boundary. A sample of data from the generated dataset can be seen in

Figure ***4***. To carry out the training and validation process, the dataset is divided into a composition of 90% for training and 10% for conducting the validation process. The dataset used to carry out

the testing process is built by pairs of data taken each from the word index so that the amount of data used to carry out the testing process is 21 pairs of data.

Table 1. Composition of Transliterated Dataset

Word Index	Word Count
A	1423
B	2090
C	1171
D	936
E	642
G	1686
H	28
I	468
J	792
K	3767
L	1494
M	4881
N	4602
O	274
P	3508
R	894
S	2943
T	2279
U	856
W	576
Y	9

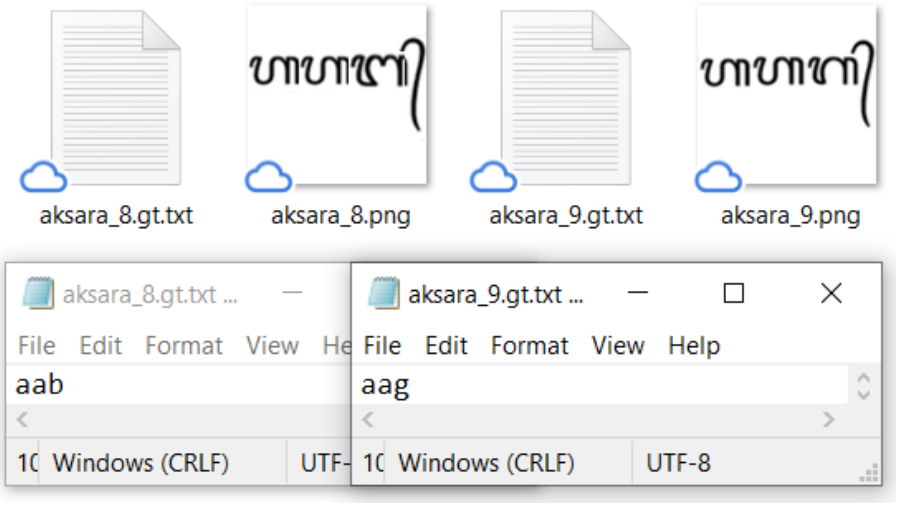


Figure 4. A Sample of Pair of Data from the Generated Dataset

3.2. Dataset Training Result

At the dataset training stage, several stages must be done to the generated dataset. The first stage groups the dataset into several groups, namely the dataset group per character, the dataset group per word, the dataset group per sentence, and the dataset group per paragraph. In the next second stage, after grouping the dataset, the datasets are arranged based on the dataset hierarchy. The preparation process of a dataset hierarchy is made into several versions and tested whether the hierarchical arrangement can increase the quality of the dataset training result. The first hierarchical arrangement of dataset training is a hierarchical arrangement by combining the dataset randomly (Random Dataset Combination Hierarchy). The percentage rate of Coincidence obtained using a random hierarchical arrangement is 25%. Next is the hierarchical

arrangement of the dataset using per character only (Single Character Dataset Combination Hierarchy). The hierarchical arrangement of this dataset gets a coincidence percentage rate of 40%. This result increased from the previous hierarchy, which consisted of a random dataset combination.

The last dataset hierarchy is a hierarchical arrangement consisting of dataset group per character, dataset group per word, dataset group per sentence, and finally, dataset group per paragraph (Combination Hierarchy of Character, Word, Sentence, and Paragraph Datasets). The hierarchical arrangement of this dataset regards the order of levels according to the order described previously. The dataset training process using this hierarchical arrangement is carried out in several training iterations until all the hierarchical levels are finished. The first level being trained is the dataset level per character. After the process is complete, it will proceed to the dataset level per word, after that the dataset level per sentence, and the last is the dataset level per paragraph. The results from the dataset training using this hierarchy got a coincidence percentage rate of 66.67%. The coincidence rate obtained has increased compared to the previous two experiments. The generated language model by the dataset training process is in the form of a trained data binary file. This language model will be the language library of the Tesseract OCR engine. Based on the result of the data training carried out, it can be seen that several dataset training scenarios were carried out with different dataset compositions and hierarchies. The result of the language model (trained data file) that will be used is the language model, which has the highest coincidence rate. The following dataset training results can be seen in
Figure ***5***.

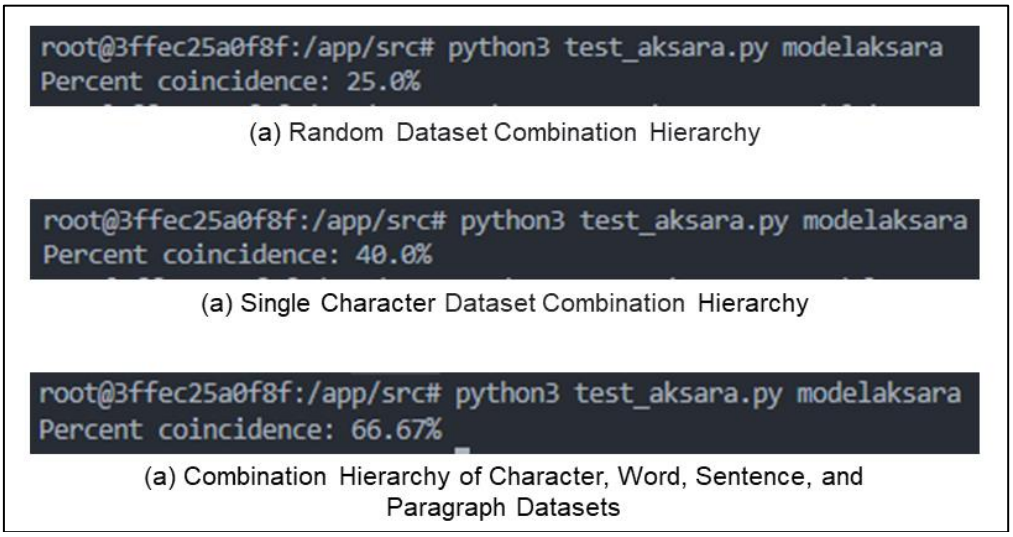


Figure 5. Dataset Training Results

The combination and the hierarchy of datasets used are the main factors influencing the increase in the coincidence performance of the three experiments conducted using different combinations of training datasets. The results of the three experiments have a common thread in terms of the hierarchical structure of the dataset. The more structured the hierarchy used, the better the coincidence rate. This increase is because Tesseract OCR learns and recognizes characters starting from the smallest unit, namely per character, then per word, after that per sentence, and finally per paragraph. The following graph of the increase in the coincidence rate can be seen in Figure 6.

Figure 6. The Coincidence Performance

The preliminary test of the resulting model language includes several test scenarios, namely the Basic Syllables test scenario, the Numerals test scenario, and the word test scenario. From the model language test process, the maximum coincidence rate was 100%, the minimum coincidence rate was 66.67%, and the average coincidence rate was 88.26%. The test results can be seen more clearly in Figure 7.

```
root@3ffec25a0f8f:/app/src# python3 test.py aksara1
Testing model language Aksara Bali
==========================================================
Scenario: Basic Syllables
Ground Truth: hanacarakadatasawalamagabangapajayanya
Output: hanacarakadatasawalamagabangapajayanya
Percent coincidence: 100.0%
----------------------------------------------------------
Scenario: Numeral
Ground Truth: 0123456789
Output: 0123456789
Percent coincidence: 100.0%
----------------------------------------------------------
Scenario: Word
Ground Truth: kala
Output: kala
Percent coincidence: 100.0%
----------------------------------------------------------
Scenario: Word
Ground Truth: paksa
Output: pakapa
Percent coincidence: 72.73%
----------------------------------------------------------
Scenario: Word
Ground Truth: raka
Output: raka
Percent coincidence: 100.0%
----------------------------------------------------------
Scenario: Word
Ground Truth: walaka
Output: walaka
Percent coincidence: 100.0%
----------------------------------------------------------
Scenario: Word
Ground Truth: krama
Output: wama
Percent coincidence: 66.67%
----------------------------------------------------------
Percent coincidence (MAX): 100.0%
Percent coincidence (MIN): 66.67%
Percent coincidence (AVG): 88.26%
==========================================================
```

Figure 7. Testing Result

3.3. Tesseract Mobile Framework Implementation

The application is built using the Flutter mobile framework by applying the concept of a clean code architecture. The clean code architecture is a blueprint for a modular system, which strictly follows a design principle called separation of concerns. More specifically, this architectural style focuses on separating the software into multiple layers to simplify the development and maintenance of the application itself. When layers are appropriately separated, code snippets can be reused, developed, and updated independently. The resulting application is also scalable, readable, testable, and can be easily maintained at any time. In addition to using clean code architecture, the application uses the Flutter Tesseract OCR dependency version 0.4.20 with a minimum SDK version of 2.12. To carry out the process of recognizing Balinese characters, the application can receive Balinese Script image input in two ways: using existing images that can be taken from the smartphone gallery or images taken from smartphone cameras. The Balinese Script image input will be processed to be recognized and converted into text. The results of the implementation of the Tesseract Mobile Framework OCR can be seen in
Figure ***8***.

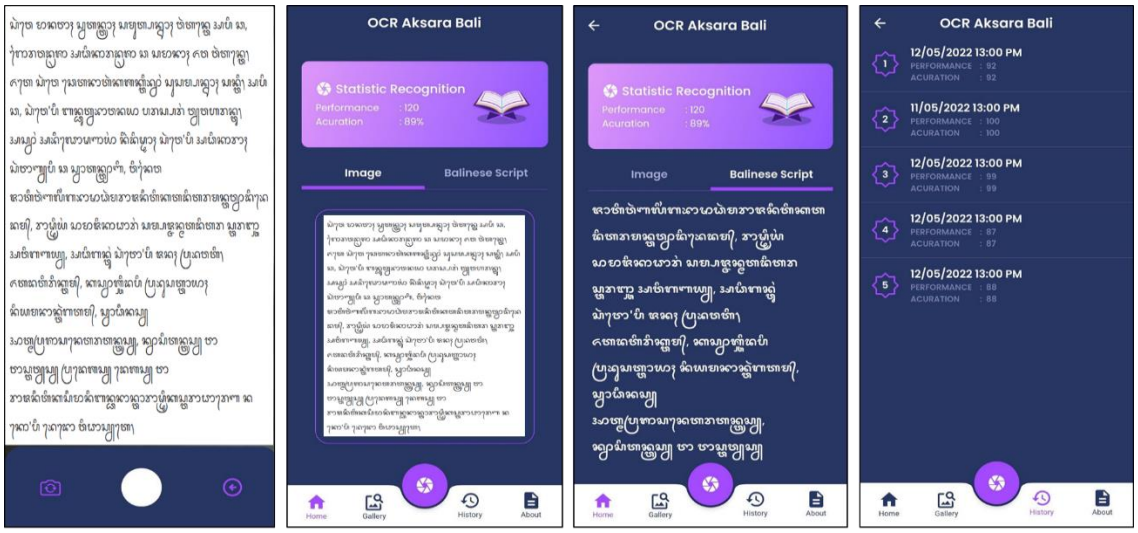


Figure 8. Balinese Script OCR Application: (a) Camera Screen; (b) Image Preview Screen; (c) Balinese Script Screen; and (d) History Screen

4. Conclusion

Several initial steps were carried out in the dataset preparation process: preparing Balinese transliteration data, converting Latin Balinese to Balinese Script using Unicode, and creating a template for the dataset generation process. Dataset generation utilizes web scraping methods and a web-based platform for the image acquisition process. The result of generating the dataset is in the form of paired files, namely a single-line-text image of Balinese characters (with "png" file extension) and its related single-line text transliteration (with "gt.txt" file extension). The dataset has been successfully generated with 35,319 image-text file pairs. The optical character recognition method and engine used in training the dataset and the Balinese character recognition process is Tesseract OCR version 5. The dataset training process consisted of three experiments with different dataset hierarchical structures. The first dataset hierarchy is a random dataset combination (Random Dataset Combination Hierarchy) which produces a coincidence rate of 25%. The second dataset hierarchy is the dataset hierarchy per character (Single Character Dataset Combination Hierarchy), with a coincidence rate of 40%. Then, the last dataset hierarchy is a combination of dataset per character, dataset per word, dataset per sentence, and dataset per paragraph (Dataset Combination Hierarchy of Character, Word, Sentence, and Paragraph) by producing a coincidence rate of 66.67%. From the three dataset hierarchical structures used for the training process, it can be concluded that the more diverse and structured the dataset

hierarchy used, the higher the coincidence rate. The training process's result from the trained data language model is then implemented into a mobile-based application platform. Mobile application development uses the Flutter mobile framework by applying a clean code architectural concept. That mobile application has several main pages: Camera Screen, Image Preview Screen, Balinese Script Screen, and History Screen. It can be concluded that generating a dataset can be a better choice when needing a large training dataset compared to some previous studies that used jTessBox tools that require relatively more time to select characters for the dataset.

Based on the results of the research process that has been carried out, it is realized that the coincidence level can still be improved. Several things are important to note to improve the coincidence rate result. In this study, the dataset used in building the language model is limited to only using synthetic data images. The next work to be carried out is to enhance several dataset hierarchies by combining several Balinese script characters with different styles, like optical characters, original data, and handwritten characters. The hierarchical arrangement of the dataset will refer to the more complex Balinese writing rules based on the existing Balinese dictionary. Furthermore, the structured hierarchy will be verified by Balinese language and script experts to ensure the validity of the dataset to be trained. Related to the image quality of the dataset, there will be stages like preprocessing, thresholding, and other image preprocessing methods before carrying out the dataset training process.

Acknowledgment

The authors gratefully acknowledge the support of the Indonesian Ministry of Education, Culture, Research, and Technology for research funding in the area of technology for information data on various forms of local wisdom.

References

[1] I. M. D. R. Mudiarta *et al.*, "Balinese character recognition on mobile application based on tesseract open source OCR engine," *Journal of Physics: Conference Series*, vol. 1516, no. 1, 2020, doi: 10.1088/1742-6596/1516/1/012017.

[2] Bali Governor, *Bali Governor Regulation No. 80 on Protection and Usage of Balinese Language, Script, and Literature*. Indonesia, 2018.

[3] A. Qaroush, A. Awad, M. Modallal, and M. Ziq, "Segmentation-based, omnifont printed Arabic character recognition without font identification," *Journal of King Saud University - Computer and Information Sciences*, Volume 34, Issue 6, Part A, 2020, doi: 10.1016/j.jksuci.2020.10.001.

[4] T. W. Ramdhani, I. Budi, and B. Purwandari, "Optical Character Recognition Engines Performance Comparison in Information Extraction," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 8, pp. 120–127, 2021, doi: 10.14569/IJACSA.2021.0120814.

[5] G. Abdul Robby, A. Tandra, I. Susanto, J. Harefa, and A. Chowanda, "Implementation of Optical Character Recognition Using Tesseract With the Javanese Script Target in Android Application," *Procedia Computer Science*, vol. 157, pp. 499–505, 2019, doi: 10.1016/j.procs.2019.09.006.

[6] H. Hassani and S. Idress, "Exploiting Script Similarities to Compensate for the Large Amount of Data in Training Tesseract LSTM: Towards Kurdish OCR," *Applied Sciences*, p. 20, Oct. 2021, doi: 10.3390/app11209752.

[7] R. Smith, "An Overview of the Tesseract OCR Engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2007, pp. 629–633, doi: 0.1109/ICDAR.2007.4376991.

[8] G. Indrawan, I. K. Paramarta, K. Agustini, and Sariyasa, "Latin-to-Balinese Script Transliteration Method on Mobile Application: A comparison," *The Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 10, no. 3, pp. 1331–1342, 2018.

[9] S. Chaudhari, R. Aparna, V. G. Tekkur, G. L. Pavan, and S. R. Karki, "Ingredient/Recipe Algorithm using Web Mining and Web Scraping for Smart Chef," *Proceedings CONECCT 2020 - 6th IEEE International Conference on Electronics, Computing and Communication Technologies*, no. 3, pp. 22–25, 2020, doi: 10.1109/CONECCT50063.2020.9198450.

[10] W. Uriawan, A. Wahana, D. Wulandari, W. Darmalaksana, and R. Anwar, "Pearson

Correlation Method and Web Scraping For Analysis of Islamic Content on Instagram Videos," *Proceedings - 2020 6th International Conference on Wireless and Telematics (ICWT) 2020*, 2020, doi: 10.1109/ICWT50448.2020.9243626.

[11] G. Adomavicius and A. Tuzhilin, "Web Scraping: State of the art," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2019.

[12] Tesseract OCR, "Tesseract User Manual," *Github*, 2018. https://tesseract-ocr.github.io/tessdoc/ (accessed Jul. 08, 2022).

[13] S. Idrees and H. Hassani, "Exploiting Script Similarities to Compensate For The Large Amount of Data In Training Tesseract LSTM: Towards Kurdish OCR," *Applied Sciences*, vol. 11, no. 20, 2021, doi: 10.3390/app11209752.

[14] P. Kumar, P. Sihag, P. Chaturvedi, K. V. Uday, and V. Dutt, "BS-LSTM: An Ensemble Recurrent Approach to Forecasting Soil Movements in the Real World," *Front. Earth Sci., 23 August 2021 Sec. Environmental Informatics and Remote Sensing*, vol. 9, no. August, pp. 1–23, 2021, doi: 10.3389/feart.2021.696792.

[15] C. Clausner, A. Antonacopoulos, and S. Pletschacher, "Efficient and effective OCR engine training," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 23, no. 1, pp. 73–88, 2020, doi: 10.1007/s10032-019-00347-8.

[16] V. K. Kaliappan, S. Yu, R. Soundararajan, S. Jeon, D. Min, and E. Choi, "High-Secured Data Communication for Cloud Enabled Secure Docker Image Sharing Technique Using Blockchain-Based Homomorphic Encryption," *Energies*, vol. 15, no. 15, 2022, doi: 10.3390/en15155544.

[17] N. H. Khan and A. Adnan, "Urdu optical character recognition systems: Present contributions and future directions," *IEEE Access*, vol. 6, pp. 46019–46046, 2018, doi: 10.1109/ACCESS.2018.2865532.

[18] K. O. Mohammed Aarif and S. Poruran, "OCR-Nets: Variants of Pre-trained CNN for Urdu Handwritten Character Recognition via Transfer Learning," *Procedia Computer Science*, vol. 171, no. 2019, pp. 2294–2301, 2020, doi: 10.1016/j.procs.2020.04.248.

[19] B. Wang, Y. W. Ma, and H. T. Hu, "Hybrid model for Chinese character recognition based on Tesseract-OCR," *International Journal of Internet Protocol Technology*, vol. 13, no. 2, pp. 102–108, 2020, doi: 10.1504/IJIPT.2020.106316.

[20] R. Bassam *et al.*, "Autonomous Assistance System for Visually Impaired using Tesseract OCR & gTTS Autonomous Assistance System for Visually Impaired using Tesseract OCR & gTTS," *Journal of Physics: Conference Series, Volume 2327, 4th International Conference on Intelligent Circuits and Systems*, doi: 10.1088/1742-6596/2327/1/012065.

[21] D. Sporici, E. Cus, and C. Boiangiu, "Using Convolution-Based Preprocessing," *SS symmetry*, 2020.

[22] Google, "Flutter architectural overview." https://docs.flutter.dev/resources/architectural-overview (accessed February 06, 2022).

[23] Google, "Dart overview." https://dart.dev/overview (accessed Feb. 06, 2022).

[24] N. Chigali, S. R. Bobba, K. Suvarna Vani, and S. Rajeswari, "OCR assisted translator," *7th International Conference on Smart Structures and Systems (ICSSS)*, July 2020, doi: 10.1109/ICSSS49621.2020.9202034.