Enhancing OCR Accuracy for Bugis Language to Bahasa Indonesia Dictionary Conversion through Image Pre-processing and Scaling Techniques

1st Mohammad Teduh Uliniansyah
Research Center for Data and Information Sciences
National Agency for Research and Innovation
Jakarta, Indonesia
m.teduh.uliniansyah@brin.go.id

2nd Agung Santosa
Research Center for Data and Information Sciences
National Agency for Research and Innovation
Jakarta, Indonesia
agung.santosa@brin.go.id

3rd Rachmawan Atmaji Perdana
Research Center for Artificial Intelligence and Cyber Security
National Agency for Research and Innovation
Jakarta, Indonesia
rach020@brin.go.id

2024 International Conference on Computer, Control, Informatics and its Applications (IC3INA) | 979-8-3315-4231-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/IC3INA64086.2024.10732090

***Abstract*—Optical Character Recognition (OCR) systems often struggle with low-resource languages like Bugis due to script complexity and limited digitized resources. This research aims to improve OCR accuracy for Bugis–Indonesian dictionary conversion by applying advanced image pre-processing techniques. We evaluated the performance of Tesseract-Vanilla, Tesseract with ImageMagick, Python PIL, and ChatGPT's OCR models by measuring Character Error Rate (CER) and Word Error Rate (WER). The results show that Tesseract with ImageMagick significantly outperformed other methods, reducing CER from 8.94% to 2.82% and WER from 35.35% to 13.54%. These findings highlight the effectiveness of pre-processing techniques in enhancing OCR performance for low-resource languages like Bugis.**

***Keywords*—OCR, Scanned Bugis – Indonesian Dictionary, Tesseract, ImageMagick, Python PIL Library, ChatGPT**

I. Introduction

The digitization of historical and regional dictionaries holds immense value in preserving and disseminating linguistic knowledge. Accurate optical character recognition is a crucial step in digitizing regional languages. Bugis language, spoken by over three and half million people primarily in South Sulawesi, Indonesia, is one such unique regional language with a rich linguistic and cultural tradition [1]. This study aims to explore various techniques to enhance the accuracy of OCR for the conversion of a Bugis language to Bahasa Indonesia dictionary, which utilizes alphanumeric characters and some multi-byte characters rather than the native Bugis script. The Bugis script, a syllabary with unique linguistic features, presents challenges not typically encountered in more common scripts like Latin or Devanagari. Unlike alphabetic scripts, Bugis characters represent syllables, and the script's cursive nature further complicates segmentation and recognition. Moreover, Bugis script has limited OCR-compatible resources, making tailored approaches crucial for improving recognition accuracy. The While considerable progress has been made in the field of OCR, with advancements in AI and machine learning techniques, regional languages like Bugis still pose unique challenges that require tailored approaches.

The key contributions of this study are threefold: 1) it investigates the effectiveness of image pre-processing and scaling techniques for enhancing OCR accuracy on a Bugis language to Bahasa Indonesia dictionary; 2) it examines the performance of combining Tesseract OCR with image processing methods using Python's PIL library; and 3) it explores the potential of leveraging ChatGPT with detailed prompting to further improve OCR accuracy for this low-resource language.

II. Related Work

Recent advancements in Optical Character Recognition (OCR) technology have aimed at improving the accuracy and adaptability of OCR systems, especially for low-resource languages and diverse applications. Existing OCR technologies have shown limitations in handling scripts that deviate from the familiar Latin alphabet, particularly when dealing with low-resource languages such as Bugis. Studies like those by Indrawan & Pramarta [2] and Chaudhuri et al. [3] have highlighted these challenges. Similarly, research by Susanto et al. [4] has examined the replicable benchmarking of neural machine translation on low-resource local languages in Indonesia, including Javanese, Sundanese, Minangkabau, and Balinese.

In recent years, there has been a significant focus on improving OCR accuracy across various domains. Studies by Salehudin [5], Lowe [6], and Toro et al. [7] emphasize the ongoing efforts to enhance OCR accuracy and adaptability, particularly in recognizing characters in engineering drawings, industrial sticker information, and document layout analysis. The integration of AI and machine learning in OCR has led to notable improvements in accuracy, as demonstrated in research by Indrawan et al. [8] and others. Techniques such as document layout analysis, text line detection, and font size optimization have been introduced to further enhance OCR accuracy [9].

Several studies have explored OCR for regional languages in the past decade. Susanto et al. examined the replicable benchmarking of neural machine translation on low-resource local languages in Indonesia, including Javanese, Sundanese, Minangkabau, and Balinese [4]. Indrawan et al. proposed a method to accommodate backward compatibility on a learning application-based transliteration to the Balinese script [10]. Indrawan & Pramarta developed a learning mobile application for Latin-to-Balinese script transliteration, achieving a high accuracy level of around 98% [8].

Furthermore, the integration of deep learning with OCR has significantly contributed to the progress of OCR technology. Research by Rao [11], Wang [12], and Li [13] delves into the historical development and future directions of OCR, emphasizing its potential to streamline data entry

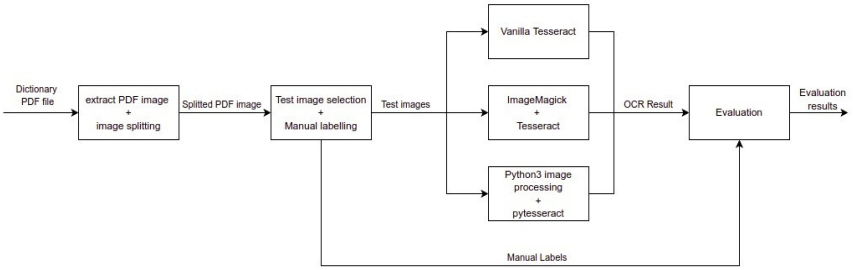

Fig. 1. Illustration of the overall OCR methodology applied in the study.

processes and improve recognition accuracy for languages like Uyghur and Asante Twi [13], [14].

While these advancements are promising, the unique characteristics of the Bugis language, such as its limited availability of digitized resources and the use of alphanumeric and multi-byte characters in the dictionary, necessitate further investigation. Studies like those by Raj et al. [15] and Singh [16] underscore the importance of tailoring OCR systems to specific languages and scripts, such as Devanagari and Urdu. The development of OCR systems for multilingual and handwritten text recognition has expanded the applicability of OCR technology [17][18].

In conclusion, the recent research landscape surrounding OCR technology reflects a concerted effort to enhance accuracy, efficiency, and adaptability across various domains. The integration of machine learning, deep learning, and advanced algorithms has propelled OCR systems to new heights, paving the way for improved recognition of diverse languages, scripts, and document types. As OCR continues to evolve, addressing challenges, refining algorithms, and exploring novel applications will be crucial in unlocking the full potential of this transformative technology.

III. Methodology

Fig. 1 illustrates the flow of methodology in this study. The following subsections provide detailed explanations for each component depicted in the figure.

A. Dataset Preparation

This study used a scanned version of the Bugis–Indonesian dictionary [19] as its raw data. Fig. 2 shows an example of a scanned page. As seen in Fig. 2, the dictionary entries were written in two columns. The first challenge we encountered was separating the first and second columns. We tried many techniques with varying results. Some produced good outcomes, but frequently the results were unsatisfactory. Therefore, we ultimately decided to crop each page image into two separate images. Some examples are shown in Fig. 3. The dictionary contains 211 pages of entries. Consequently, we ended up with 422 images containing the dictionary entries. From the total 422 scanned dictionary pages, we selected 20 files randomly to represent the dataset. Random selection was performed to ensure unbiased sampling and to maintain diversity in terms of font clarity, image quality, and page layout. This selection strategy is meant to cover a broad range of potential OCR issues, such as varying degrees of image noise, text skew, and inconsistent lighting. Although the random sampling might not represent all possible variations, the sample size was deemed sufficient for proof of concept. Future studies could expand the dataset to validate the findings on a larger scale.

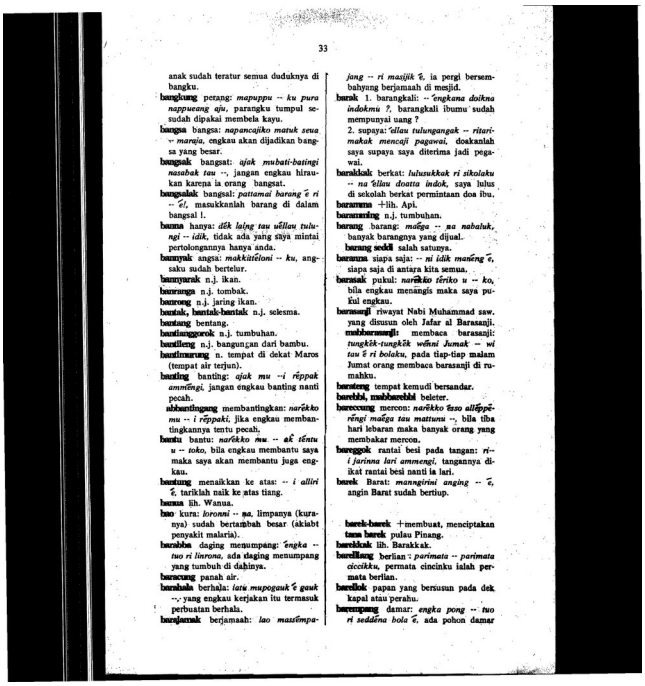

Fig. 2. Example of a scanned Bugis-Indonesian dictionary page before processing.

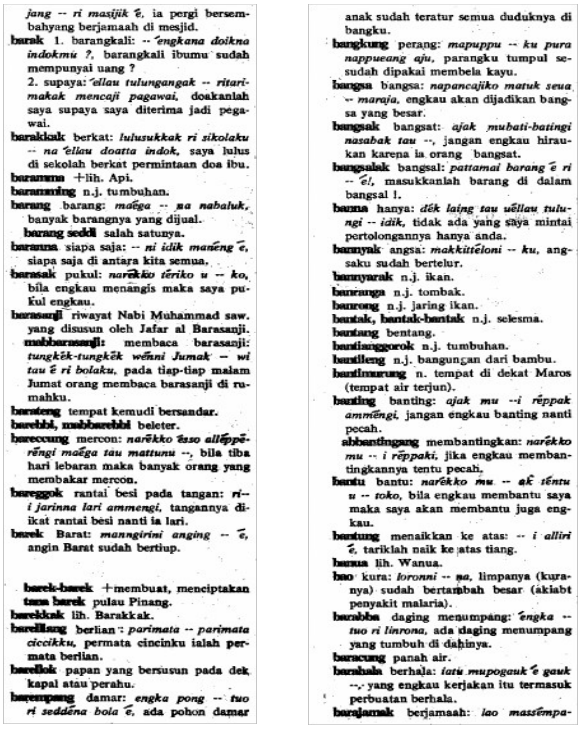

Fig. 3. The condition of some images after cropping

B. OCR Tools and Techniques

We used Tesseract version 4.1.1 as the main OCR tool to extract text from the image files [20]. For image processing, we used ImageMagick [21] and the Python Imaging Library (PIL Library). The image processing techniques applied were: converting the image to grayscale, scaling the image by 1.5 times, changing the DPI of the image to 300, deskewing the image, and enhancing the image (including histogram stretching, applying a Gaussian blur with a radius of 38, applying an unsharp mask with a radius of 11, and applying a selective blur with a threshold of 1). For processing the image files with ImageMagick, we used the 'convert' tool [22] with the command '*convert -units PixelsPerInch inFile -colorspace Gray -resize 150% -density 300 outFile*' and the 'textcleaner' script [23] with the following command: '*textcleaner -g -e stretch -f 38 -o 8 -s 1 inFile outFile*'.

C. Experimental Setup

The experimental setup was as follows:

1. Tesseract vanilla.

Tesseract converted the image file without any image preprocessing.

2. Tesseract with ImageMagick.

The image files were pre-processed with the previously mention image processing techniques using ImageMagick script, and the results were given to Tesseract.

3. Tesseract with PIL Library.

The image files were pre-processed with the previously mention image processing techniques using Python PIL Library, and the results were given to Tesseract.

4. ChatGPT

We used ChatGPT's OCR [24] with tailored prompt as follows: "As an expert in Bugis language and Bahasa Indonesia, please extract texts from this scanned image of a page from a Bugis to Bahasa Indonesia dictionary. Format the output as text that have the same layout as shown in the image, do not merge lines and do not add any new words, base the output only from the scanned image. Enlarge the image to 300dpi and perform any necessary image enhancement to get the best result".

5. Post-processing

We also conducted experiments using several post-processing steps, which mainly involved changing specific character patterns. The post-processing was done by employing sed tools [25], which were primarily used to modify certain characters. The sed commands used are as follows:

sed -i 's/^\‘ //g' $outFile (remove character ‘ in the beginning of a line)

sed -i 's/“/"/g' $outFile (change character “ to ")

sed -i 's/€/é/g' $outFile (change character € to é)

sed -i 's/@/é/g' $outFile (change character @ to é)

sed -i 's/^% /é, /g' $outFile (change character % in the beginning of a line to é)

sed -i 's/¢/é/g' $outFile (change character ¢ to é)

sed -i 's/&/é/g' $outFile (change character & to é)

sed -i 's/—/--/g' $outFile (change character — to --)

sed -i 's/~/--/g' $outFile (change character ~ to --)

sed -i 's/\. °\$/\./g' $outFile (change a character string of ., space, and ° to . and a space.

sed -i 's/\. [A-Z]\$/\./g' $outFile (change a character string of . and a capital letter to a dot.

sed -i 's/\. \.\$/\./g' $outFile (change a character string of a dot, a space, and a dot to a dot.

sed -i 's/^[\"\:]//g' $outFile (remove character " or : in the beginning of line)

sed -i 's/ - / -- /g' $outFile (remove a character string of a space, -, and a space to a character string of a space and --.

sed -i 's/ ‘/ /g' $outFile (remove a character string of a space and ‘ to a space.

sed -i 's/ = / -- /g' $outFile (remove a character string of a space, =, a space to a character string of a space, --, and a space.

sed -i 's/-\r//g' $outFile (remove a character string of - and a newline)

D. Evaluation

Using the metric values of both the Character Error Rate (CER) and Word Error Rate (WER) for all twenty image files, we calculated the standard deviation values and assessed statistical significance using p-values.

The CER and WER were calculated as follows:

$$CER = \frac{(Insertions + Deletions + Substitutions)}{Number\ of\ Characters} \quad (1)$$

$$WER = \frac{(Insertions + Deletions + Substitutions)}{Number\ of\ Words} \quad (2)$$

These metrics provide insight into how well the OCR system performs at the character and word level, respectively, with lower values indicating higher accuracy.

Insertions, deletions, and substitutions are standard metrics used in OCR to measure discrepancies between the predicted and actual text.

- Insertions refer to extra characters added by the OCR system.

- Deletions are characters that were omitted from the recognized text

- Substitutions occur when an incorrect character is recognized in place of the correct one.

These errors arise from inherent challenges in recognizing complex layouts and character variations in scanned images, even before post-processing.

IV. Results and Discussions

Table I provides an overview of the OCR performance across twenty image files. The key metrics include the total number of characters/words, the number of correct recognitions, and the error rates (both in percentage and absolute terms). The data clearly illustrates that Tesseract with ImageMagick consistently performs better with the lowest error rates, while the vanilla Tesseract has the highest error rates.

Table II provides detailed performance metrics for a single image file (33a). We selected the 33a file for Table II because it represents typical OCR challenges present across the dataset, such as noise and skewed text. By focusing on this file, we provide a clear and detailed comparison of performance metrics like CER and WER, without overwhelming the reader. While aggregate data for all files is

TABLE I. COMBINED OCR PERFORMANCE METRICS FOR 20 IMAGE FILES

Types of Accuracy Measurement	Methods	Total Chars/Words	Corrects	Corrects (%)	Errors	Errors (%)	Insertions	Deletions	Substitutions
Character Error Rate (CER) with no post-processing	Tesseract-Vanilla	29,915	27,242	91.06	2,673	8.94	362	865	1,446
	Tesseract-ImageMagick	29,915	**28,839**	**96.40**	**1,076**	**3.60**	298	156	622
	Tesseract-Python (PIL Library)	29,915	28,750	96.11	1,165	3.89	200	224	741
	ChatGPT	29,915	28,537	95.39	1,378	4.61	298	257	823
Character Error Rate (CER) with post-processing	Tesseract-Vanilla	29,915	27,564	92.14	2,351	7.86	368	790	1,193
	Tesseract-ImageMagick	29,915	**29,071**	**97.18**	**844**	**2.82**	175	186	483
	Tesseract-Python (PIL Library)	29,915	28,859	96.47	1,056	3.53	177	262	617
	ChatGPT	29,915	28,636	95.72	1,279	4.28	257	312	710
Word Error Rate (WER) with no post-processing	Tesseract-Vanilla	4,854	3,138	64.65	1,716	35.35	14	131	1,571
	Tesseract-ImageMagick	4,854	**4,069**	**83.83**	**785**	**16.17**	69	30	686
	Tesseract-Python (PIL Library)	4,854	4,018	82.78	836	17.22	25	47	764
	ChatGPT	4,854	3,992	82.24	862	17.76	50	44	768
Word Error Rate (WER) with post-processing	Tesseract-Vanilla	4,854	3,300	67.99	1,554	32.01	8	166	1,380
	Tesseract-ImageMagick	4,854	**4,197**	**86.46**	**657**	**13.54**	27	57	573
	Tesseract-Python (PIL Library)	4,854	4,096	84.38	758	15.62	14	75	669
	ChatGPT	4,854	4,060	83.64	794	16.36	29	72	693

TABLE II. OCR PERFORMANCE METRICS FOR INDIVIDUAL IMAGE FILE (33A)

Types of Accuracy Measuremets	Image File	Methods	Total Chars/Words	Corrects	Corrects (%)	Errors	Errors (%)	Insertions	Deletions	Substitutions
Character Error Rate (CER) with no post-processing	33a	Tesseract-Vanilla	1,506	1,244	82.60	262	17.40	19	100	143
		Tesseract-ImageMagick	1,506	**1,419**	**94.22**	**87**	**5.78**	8	15	64
		Tesseract-Python (PIL Library)	1,506	1,410	93.63	96	6.37	9	11	76
		ChatGPT	1,506	1,402	93.09	104	6.91	8	8	88
Character Error Rate (CER) with post-processing	33a	Tesseract-Vanilla	1,506	1,263	83.86	243	16.14	17	92	134
		Tesseract-ImageMagick	1,506	**1,438**	**95.48**	**68**	**4.52**	7	10	51
		Tesseract-Python (PIL Library)	1,506	1,421	94.36	85	5.64	8	10	67
		ChatGPT	1,506	1,408	93.49	98	6.51	9	10	79
Word Error Rate (WER) with no post-processing	33a	Tesseract-Vanilla	242	96	39.67	146	60.33	2	18	126
		Tesseract-ImageMagick	242	**178**	**73.55**	**64**	**26.45**	2	4	58
		Tesseract-Python (PIL Library)	242	173	71.49	69	28.51	2	3	64
		ChatGPT	242	171	70.66	71	29.34	1	1	69
Word Error Rate (WER) with post-processing	33a	Tesseract-Vanilla	242	104	42.98	138	57.02	1	18	119
		Tesseract-ImageMagick	242	**189**	**78.10**	**53**	**21.90**	1	4	48
		Tesseract-Python (PIL Library)	242	180	74.38	62	25.62	1	3	58
		ChatGPT	242	175	72.31	67	27.69	1	1	65

shown in other tables, 33a allows for a more focused and illustrative example of the improvements from different OCR methods and tools.The patterns observed are consistent with those in Table 1, confirming the superior performance of Tesseract with ImageMagick. The ChatGPT method also shows promising results, though it is slightly outperformed by Tesseract with ImageMagick.

Tables III and IV provide aggregate results across all 20 image files. These tables summarize the mean and standard deviation of OCR performance metrics, offering insights into the variability and reliability of the different methods across the entire dataset. A lower standard deviation, as observed for Tesseract with ImageMagick, indicates more consistent performance, which is desirable in practical applications.The vanilla Tesseract, on the other hand, shows higher variability, suggesting inconsistent performance across different images. p-values in Table IV indicate the statistical significance of differences in OCR performance between various methods. The extremely low p-values (<0.05) between Tesseract-Vanilla and the other methods suggest that the differences are statistically significant, particularly when post-processing is applied. This reinforces the conclusion that methods like Tesseract+ImageMagick with post-processing offer significantly better performance.

Tesseract-Vanilla method, while being a baseline OCR approach, shows substantial room for improvement, especially in scenarios where accuracy is critical. The standard deviation values suggest that this method is less reliable, as its performance fluctuates significantly across different images (Table III). This inconsistency could lead to challenges in environments where consistent output quality is required.Tesseract-ImageMagick emerges as a robust enhancement over the vanilla version, showing not only a lower mean CER and WER but also reduced variability. This indicates that the ImageMagick preprocessing step effectively standardizes the input images, leading to more

TABLE III. MEAN AND STANDARD DEVIATION OF OCR PERFORMANCE METRICS

Character Error Rate

Methods	Mean	StdDev	Min	Max
Tesseract-Vanilla	8.90%	0.0411	4.75%	22.12%
Tesseract-ImageMagick	3.59%	0.0126	1.58%	6.81%
Tesseract-Python (PIL Library)	3.88%	0.0146	1.51%	7.13%
ChatGPT	4.55%	0.0316	1.51%	15.77%
Tesseract-Vanilla+Post-Proc	7.82%	0.0401	4.28%	20.81%
Tesseract-ImageMagick+Post-Proc	**2.81%**	0.0110	1.25%	6.09%
Tesseract-Python (PIL Library)+Post-Proc	3.52%	0.0124	1.78%	6.15%
ChatGPT+Post-Proc	4.24%	0.0318	1.31%	15.89%

Word Error Rate

Methods	Mean	StdDev	Min	Max
Tesseract-Vanilla	35.33%	0.1154	23.60%	69.83%
Tesseract-ImageMagick	16.16%	0.0546	8.30%	32.64%
Tesseract-Python (PIL Library)	17.23%	0.0575	8.00%	30.58%
ChatGPT	17.69%	0.0636	7.79%	30.17%
Tesseract-Vanilla+Post-Proc	31.96%	0.1125	21.20%	65.29%
Tesseract-ImageMagick+Post-Proc	**13.50%**	0.0501	7.79%	29.75%
Tesseract-Python (PIL Library)+Post-Proc	15.60%	0.0513	9.84%	26.86%
ChatGPT+Post-Proc	16.30%	0.0616	6.97%	27.91%

TABLE IV. P-VALUES OF OCR PERFORMANCE METRICS

P-Values CER

Methods	Tesseract-Vanilla	Tesseract-ImageMagick	Tesseract-Python (PIL Library)	ChatGPT	Tesseract-Vanilla +Post-Proc	Tesseract-ImageMagick +Post-Proc	Tesseract-Python (PIL Library) +Post-Proc	ChatGPT +Post-Proc
Tesseract-Vanilla	-	3.81E-07	5.54E-07	6.75E-05	6.83E-11	1.00E-07	5.44E-07	5.04E-05
Tesseract-ImageMagick		-	**7.01E-02**	**1.42E-01**	5.82E-06	4.54E-08	**6.11E-01**	**3.31E-01**
Tesseract-Python (PIL Library)			-	**3.06E-01**	1.03E-05	1.74E-05	2.54E-04	**5.96E-01**
ChatGPT				-	1.03E-03	1.18E-02	**1.12E-01**	2.89E-04
Tesseract-Vanilla+Post-Proc					-	1.07E-06	7.69E-06	6.72E-04
Tesseract-ImageMagick+Post-Proc						-	1.28E-04	3.85E-02
Tesseract-Python (PIL Library)+Post-Proc							-	**2.72E-01**
ChatGPT+Post-Proc								-

P-Values WER

Methods	Tesseract-Vanilla	Tesseract-ImageMagick	Tesseract-Python (PIL Library)	ChatGPT	Tesseract-Vanilla +Post-Proc	Tesseract-ImageMagick +Post-Proc	Tesseract-Python (PIL Library) +Post-Proc	ChatGPT +Post-Proc
Tesseract-Vanilla	-	2.41E-10	5.95E-10	3.77E-09	2.42E-09	1.40E-10	1.32E-09	6.01E-09
Tesseract-ImageMagick		-	**1.24E-01**	**6.28E-02**	3.68E-09	1.37E-06	**4.59E-01**	**8.84E-01**
Tesseract-Python (PIL Library)			-	**5.86E-01**	9.23E-09	2.00E-04	7.04E-04	**3.51E-01**
ChatGPT				-	4.89E-08	3.47E-05	1.40E-02	1.63E-03
Tesseract-Vanilla+Post-Proc					-	1.09E-09	1.21E-08	5.87E-08
Tesseract-ImageMagick+Post-Proc						-	7.86E-03	4.40E-03
Tesseract-Python (PIL Library)+Post-Proc							-	**3.82E-01**
ChatGPT+Post-Proc								-

reliable OCR performance. The statistical significance of these improvements is confirmed by the low p-values (Table IV), which suggest that the observed differences are not due to random chance.

A. *Impact of Post-Processing*

Post-processing techniques, as applied in the Tesseract-this, showing statistically significant improvements when post-processing is applied.

B. *ChatGPT's Performance*

The ChatGPT method, while slightly lagging behind Tesseract-ImageMagick, still provides competitive results, especially considering its flexibility in handling various text recognition tasks. The results suggest that ChatGPT could be a valuable tool in situations where traditional OCR methods

might struggle, such as with non-standard fonts or complex layouts.

C. Statistical Significance

The p-values in Table IV offer strong evidence that the observed differences in OCR performance are not merely due to random variation but are statistically significant. This is particularly relevant for industries where precision is critical, such as in digitizing historical documents or processing forms with legal or financial data. In such cases, selecting an OCR method with statistically superior performance can lead to substantial cost savings and reduced error rates.

D. Comparison with Previous Research

Our results show significant OCR accuracy improvements for Bugis using Tesseract with ImageMagick, reducing CER from 8.94% to 2.82%. Similar advancements were seen in the work of Indrawan & Pramarta [2], who achieved 98% accuracy in Latin-to-Balinese transliteration, and Susanto et al. [4], who applied neural machine translation to low-resource languages. While their focuses differ, all demonstrate the effectiveness of tailored approaches for regional languages.

In line with Salehudin's [5] study on image degradation, our pre-processing techniques similarly boosted OCR accuracy. The reduction in WER from 35.35% to 13.54% underscores the value of image pre-processing in enhancing performance, particularly for low-resource languages like Bugis. This comparison validates the effectiveness of our method within the context of prior research.

V. Conclusions

Our study demonstrates the importance of tailored image pre-processing techniques in enhancing OCR accuracy for the Bugis–Indonesian dictionary. Tesseract with ImageMagick showed the best overall performance, reducing CER by approximately 6.12% and WER by 21.81% compared to the baseline Tesseract-Vanilla. While ChatGPT provided competitive results, particularly in complex recognition tasks, it was slightly outperformed by traditional OCR methods with pre-processing.

In summary, the extended analysis reaffirms the superiority of Tesseract with ImageMagick in traditional OCR tasks, while also highlighting the potential of modern AI-driven methods like ChatGPT in more complex or non-standard scenarios. The detailed statistical evaluation offers a strong foundation for making informed decisions about OCR tool selection based on the specific requirements of a project.

Future work could explore hybrid approaches that combine the strengths of Tesseract (with pre-processing) and advanced models like ChatGPT to further reduce error rates and improve reliability. Additionally, future efforts should expand the dataset to include more images to validate the findings of this study on a larger scale..

References

[1] A. R. Talaohu, "The phonetics system in Buginese language," KLAUSA (Kajian Linguistik Pembelajaran Bahasa Dan Sastra), vol. 3, no. 01, pp. 23–44, 2019. doi: 10.33479/klausa.v3i01.146.

[2] G. Indrawan and I. K. Pramarta, "The Development of Learning Mobile Application of Latin-to-Balinese Script Transliteration," JPKM, vol. 4, no. 2, pp. 123-123, Oct. 2019. doi: 10.30818/jpkm.2019.2040202.

[3] B. Chaudhuri, U. Pal, and M. Mitra, "Automatic recognition of printed Oriya script," Springer Science+Business Media, vol. 27, no. 1, pp. 23-34, Feb. 2002. doi: 10.1007/bf02703310.

[4] L. Susanto, R. Diandaru, A. Krisnadhi, A. Purwarianti, and D. Wijaya, "Replicable Benchmarking of Neural Machine Translation (NMT) on Low-Resource Local Languages in Indonesia," Cornell University, Jan. 2023. doi: 10.48550/arxiv.2311.00998.

[5] M. Salehudin, "Analysis of optical character recognition using easyocr under image degradation," Journal of Physics Conference Series, vol. 2641, no. 1, 012001, 2023. doi: 10.1088/1742-6596/2641/1/012001.

[6] M. Lowe, "Advancing machine learning with ocr2seq: an innovative approach to multi-modal data augmentation," Journal of Big Data, vol. 11, no. 1, 2024. doi: 10.1186/s40537-024-00927-4.

[7] J. Toro, A. Wiberg, and M. Tarkian, "Optical character recognition on engineering drawings to achieve automation in production quality control," Frontiers in Manufacturing Technology, vol. 3, 2023. doi: 10.3389/fmtec.2023.1154132.

[8] G. Indrawan, N. N. H. Puspita, I. K. Paramarta, and S. Sariyasa, "LBtrans-Bot: A Latin-to-Balinese Script Transliteration Robotic System based on Noto Sans Balinese Font," Institute of Advanced Engineering and Science (IAES), vol. 12, no. 3, pp. 1247-1247, Dec. 2018. doi: 10.11591/ijeecs.v12.i3.pp1247-1256.

[9] A. Fateh, "Enhancing optical character recognition: efficient techniques for document layout analysis and text line detection," Engineering Reports, 2023. doi: 10.1002/eng2.12832.

[10] G. Indrawan, I. K. Paramarta, I. G. Nurhayata, and S. Sariyasa, "A Method to Accommodate Backward Compatibility on the Learning Application-based Transliteration to the Balinese Script," Science and Information Organization, vol. 12, no. 6, Jan. 2021. doi: 10.14569/ijacsa.2021.0120631.

[11] X. Rao, "Deep-learning-based annotation extraction method for chinese scanned maps," Isprs International Journal of Geo-Information, vol. 12, no. 10, p. 422, 2023. doi: 10.3390/ijgi12100422.

[12] J. Wang, "A study of the ocr development history and directions of development," Highlights in Science Engineering and Technology, vol. 72, pp. 409-415, 2023. doi: 10.54097/bm665j77.

[13] W. Li, "A three-stage uyghur recognition model combining the attention mechanism and different convolutional recurrent networks," Applied Sciences, vol. 13, no. 17, p. 9539, 2023. doi: 10.3390/app13179539.

[14] K. Drah, "Handwritten character recognition using convolutional neural network for asante twi language alphabets," Research Square, 2023. doi: 10.21203/rs.3.rs-3638464/v1.

[15] A. Raj, S. Sharma, J. Singh, and A. Singh, "Revolutionizing data entry: an in-depth study of optical character recognition technology and its future potential," International Journal for Research in Applied Science and Engineering Technology, vol. 11, no. 2, pp. 645-653, 2023. doi: 10.22214/ijraset.2023.49108.

[16] M. Singh, "Ocr for devanagari script," Matec Web of Conferences, vol. 392, p. 01128, 2024. doi: 10.1051/matecconf/202439201128.

[17] V. Neerugatti, "Multi-lingual character recognition and extraction using recurrent neural networks," I-Manager's Journal on Image Processing, vol. 10, no. 4, p. 1, 2023. doi: 10.26634/jip.10.4.20293.

[18] K. Jaiswal, A. Suneja, A. Kumar, A. Ladha, and N. Mishra, "Preprocessing low quality handwritten documents for ocr models," International Journal for Research in Applied Science and Engineering Technology, vol. 11, no. 4, pp. 2980-2985, 2023. doi: 10.22214/ijraset.2023.50664.

[19] M. Ide Said, Bugis – Indonesian Dictionary (In Indonesian: Kamus Bahasa Bugis-Indonesia). Pusat Pembinaan dan Pengembangan Bahasa eBooks, 1977. Available: http://repositori.kemdikbud.go.id/2856.

[20] R. Smith, "An Overview of the Tesseract OCR Engine," Proceedings of the International Conference on Document Analysis and Recognition, 2007. doi: 10.1109/icdar.2007.4376991.

[21] K. Takahashi, "ImageMagick," The Journal of the Institute of Image Information and Television Engineers, vol. 61, no. 12, pp. 1725–1728, 2007. doi: 10.3169/itej.61.1725.

[22] "ImageMagick – Command-line Tools: Convert," ImageMagick. https://imagemagick.org/script/convert.php

[23] F. Weinhaus, "Fred's ImageMagick Scripts: TEXTCLEANER." Available: http://www.fmwconcepts.com/imagemagick/textcleaner/.

[24] ChatGPT, "ChatGPT." Available: https://chatgpt.com/g/g-L29PpDmgg-ocr-formerly-chatocr.

[25] GNU Sed - GNU Project - Free Software Foundation. Available: https://www.gnu.org/software/sed/.