JOIV : Int. J. Inform. Visualization, 8(3) - September 2024 1460-1468

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Transliterating Javanese Script Images to Roman Script Using Convolutional Neural Network with Transfer Learning

Mohammad Farid Naufal [a,*], Joko Siswantoro [a], Juan Timothy Soebroto [a]

[a] *Department of Informatics, Faculty of Engineering, Universitas Surabaya, Surabaya, Indonesia*
*Corresponding author: *faridnaufal@staff.ubaya.ac.id*

***Abstract*—The Javanese script holds immense cultural significance within Indonesia despite its diminishing usage in contemporary contexts. Its presence remains notable in specific regions of Java and remains integral to many historical documents and texts. Consequently, there is an urgent need for a transliteration system adept at converting Javanese script into contemporary scripts like Roman or Indonesian, thereby contributing to preserving Java's linguistic and cultural legacy. However, reading or transliterating Javanese script can be time-consuming, especially for longer texts, presenting considerable challenges for non-native readers. This study aims to develop an effective transliteration system for converting Javanese script into Roman script. This system addresses the pressing need to preserve Java's linguistic and cultural heritage by facilitating the readability and accessibility of Javanese script, especially for non-native readers. This study introduces an Optical Character Recognition (OCR) system tailored to identify Javanese script characters and transcribe them into Roman characters, explicitly focusing on fundamental *nglegena* and *sandhangan swara* characters. Individual characters are isolated by leveraging horizontal and vertical projection techniques, facilitating subsequent classification using a Convolutional Neural Network (CNN) employing transfer learning methodologies. The system's achievement of an impressive average similarity score of 90.78% is noteworthy, with the Xception architecture demonstrating superior efficiency in transliteration tasks. Implementing such a system harbors significant promise in safeguarding the Javanese script and enhancing its accessibility to a broader audience. This research contributes substantially to preserving and propagating Indonesia's rich cultural and linguistic heritage amidst the digital age.**

***Keywords*—Javanese script; transliteration; optical character recognition; convolutional neural network.**

Manuscript received 19 Feb. 2024; revised 19 May 2024; accepted 9 Jun. 2024. Date of publication 30 Sep. 2024.

I. INTRODUCTION

Indonesia, known for its rich cultural diversity, includes Java Island, a vibrant hub of tradition and history. Java Island is celebrated for its intricate cultural heritage, particularly highlighted by the prominence of the Javanese language and script [1], [2]. However, the Javanese script is declining and is now only used in historical documents and letters [3]–[5]. Translating Javanese script into Roman script can enhance cross-cultural understanding and communication. This system can make Javanese text more accessible and readable to non-Javanese speakers, facilitating Javanese language study and cultural preservation due to the complexity of Javanese script. Many individuals encounter challenges writing it. Some cannot recognize or decipher Javanese script characters [6], [7].

Digital images of Javanese texts can be used to develop such a system, significantly impacting language learning and preservation [8]. It would make Javanese texts more accessible to a broader audience and contribute to our understanding of Javanese language, culture, and history [9]. A system that transliterates Javanese script to Roman script using digital images would be valuable for language learning, cultural preservation, and academic research, enabling greater understanding and appreciation of Javanese language and culture.

Research on Javanese writing classification using CNN previously existed. Dewa et al. [10] created software that uses a CNN model for recognizing handwritten Javanese characters. The performance of the software was evaluated using k-fold cross-validation to measure classification accuracy and training time. The results showed that the CNN model outperformed the MLP model in accuracy but required more training time. However, the accuracy of the CNN model did not reach 90% due to the limited size of the dataset. They recommend optimizing the CNN model for a larger dataset in future work. It is important to note that the model can only classify a single character in an image, not multiple characters.

Widiarti et al. [11] proposed a model that converts Javanese script images to Latin text, comprising three sub-models: manuscript image segmentation, Javanese script image to Latin script transliteration, and syllable grouping. An adaptive approach addresses diversity issues in Javanese script image segmentation. The model uses a structural approach to extract Javanese script features using specific physical characteristics. A Javanese word dictionary is utilized to validate syllable grouping results, and a Markov chains tool is used to handle errors in the script replacement process. However, the proposed model was tested on only one Javanese manuscript, with an accuracy of 79.6%, indicating that it may be less effective for other Javanese manuscripts with different handwriting styles or variations. Additionally, the research only used the KNN algorithm with K=1 (1-NN) to classify Javanese characters. The 1-NN algorithm is prone to overfitting, which means that the model may memorize the training data instead of learning general patterns that can be applied to new data. Consequently, this may result in poor performance on unseen data.

Ilham et al. [12] utilized a segmentation technique by combining projection profile and connected component labeling. The classification approach employed a Convolutional Neural Network. The test data consisted of 20 basic handwriting Javanese characters captured by a smartphone camera. The study achieved a 90% accuracy rate in the character segmentation phase. Thus, the projection profile and connected component labeling methods can effectively perform image segmentation. During testing, the CNN method achieved an accuracy rate of 80% using 20 test images, demonstrating its effectiveness in character recognition. The limitation of this study is that it only uses 20 basic Javanese characters and has yet to be applied to *sandang swara*. Moreover, training a model from scratch on a small dataset can lead to overfitting. This means that the model may memorize the training data rather than learning general patterns that can be applied to new data.

Wibowo et al. [6] propose a comprehensive system for Javanese character recognition, integrating a feature extraction method and KNN for classification. The system encompasses various stages: character segmentation to isolate individual Javanese characters from input images, feature extraction utilizing Shape Energy and an Improvement Method with angle multipliers (10, 20, 30, 40 degrees), k-nearest Neighbor (k-NN) classification with different k values (1, 3, 5, 7) to assess performance, cross-validation sampling for testing model reliability, and performance testing on rotated images to evaluate the feature extraction method's robustness. The potential limitation arises from the use of KNN for classification, which tends to be prone to overfitting, particularly in complex datasets. KNN relies heavily on local similarities, so it may excessively fit the training data, resulting in poor generalization to unseen data.

Robby et al. [4] addressed the challenge of recognizing Javanese characters using Optical Character Recognition (OCR) methods. They collected a dataset of 5880 Javanese characters and trained multiple models using Tesseract OCR tools. Implementing these models on Android-based mobile phones, they explored two annotation methods for bounding boxes: one with separate boxes for the main body and phonetic symbols and the other with a unified box. Their analysis revealed that combining a single boundary box for the entire character, along with separate boxes for the main body and *sandangan* parts, yielded the highest accuracy. This study underscores the intricacies of developing OCR systems for non-Latin scripts like Javanese, emphasizing the importance of dataset curation, model training, and annotation methodologies in achieving accurate recognition. One potential limitation of Robby et al.'s approach to Javanese character recognition using OCR methods is the dependency on accurately annotated bounding boxes, which can be mitigated by employing Javanese transliteration techniques through transfer learning and projection profile segmentation to automate segmentation and enhance model generalization.

The previous studies related to Javanese script classification and transliteration have their limitations. Dewa et al. [10] achieved suboptimal accuracy due to the limited dataset size and the inability to classify multiple characters in a single image. Widiarti et al. [11], the model was tested on only one manuscript and used the 1-NN algorithm, which is prone to overfitting. Wibowo et al.'s [6] method also utilizes the KNN algorithm for classification, which tends to exhibit overfitting tendencies. Ilham et al. [12] had a small dataset and did not include *sandhang swara*, with training from scratch leading to potential overfitting. This study uses CNN with transfer learning combined with projection profile segmentation—the dependency on accurately annotated bounding boxes, as highlighted by Robby et al. [4], further emphasizes the need for automated segmentation techniques like projection profile segmentation to improve model robustness and generalization in Javanese character recognition tasks.

Combining CNN with transfer learning and projection profile segmentation can solve the overfitting problem because transfer learning has a pre-trained model that has already learned general patterns from a large dataset, making it more suitable for a small dataset like this study. Additionally, this study was able to transliterate multiple characters in the Javanese script by using projection profile segmentation to obtain every character in one image, including *sandhangan swara* characters, enabling the model to classify all characters simultaneously. This approach is more efficient and accurate than classifying a single character in an image. Therefore, using CNN with transfer learning and projection profile segmentation in this study can overcome the limitations of previous studies and provide more accurate and reliable results in Javanese script transliteration.

II. Materials and Methods

The method starts with preprocessing, which includes gray scaling, Gaussian blurring, thresholding, and dilation to enhance the image quality and improve the accuracy of character segmentation. The next step is segmentation, which uses projection profile segmentation to separate individual Javanese characters from the input image. The separated characters are then classified using transfer learning with three pre-trained models: MobileNetV2, VGG-16, and Xception. Finally, the performance of the proposed methodology is evaluated in terms of the F1 score. Figure 1 depicts the method of this study.

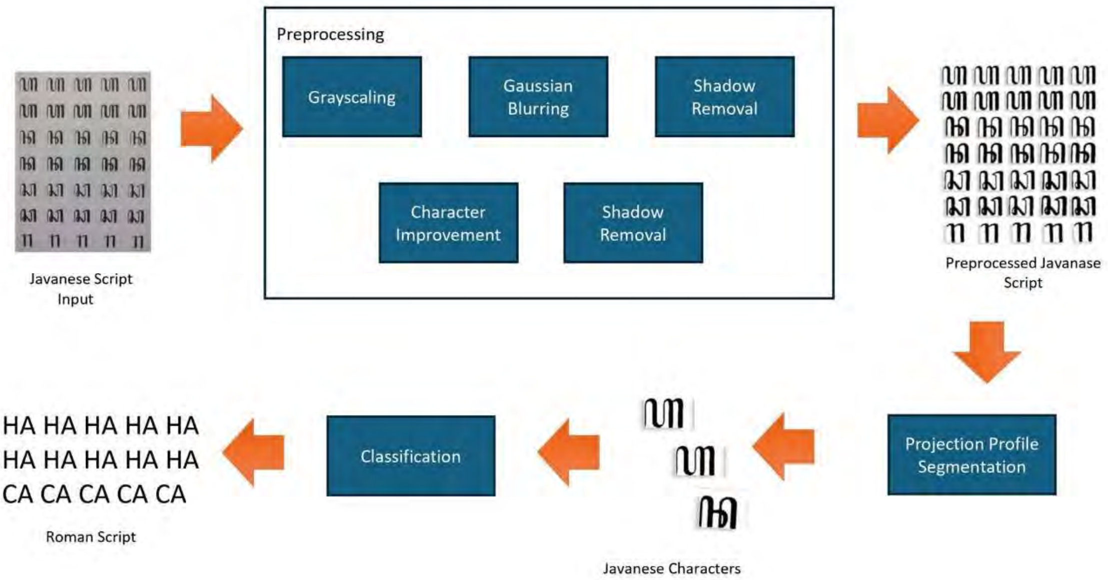


Fig. 1 Method proposed in this study

A. Dataset

This study focuses on the Javanese script, specifically basic Javanese characters (*nglegena*) and the symbol or marker that will change the base vowel (*swara sandhangan*), comprising 46 classes. It consists of 20 *nglegena* characters, 20 *nglegena* characters combined with *swara sandhangan* suku (pronounced as "u") because the *sandhangan suku* combines directly with the *nglegena* character, 2 special characters for *"re"* and *"le"* because they cannot be created by combining *nglegena* characters *"ra"* and *"la"* with *pepet sandhangan*, and 4 other *sandhangan swara* characters*: wulu, pepet, taling, and talung*. For each class, 40 images are provided, divided into 2 types: using the "*hanacaraka*" font and the second type using the "*hanacaraka*" font with bold attributes. Table 1 shows 20 nglegena along with "re" and "le" characters and their pronunciation using Roman characters. Table 2 shows the *sandhangan swara wulu, suku, pepet, taling*, and *talung*. Please note that the *sandangan suku,* when pronounced as "u," is written continuously along with the *nglegena* character.

TABLE I
NGLEGENA CHARACTERS ALONG WITH THEIR PRONUNCIATION USING ROMAN CHARACTER

Javanese Character	Roman Character	Javanese Character	Roman Character
	ha		da
	na		ja
	ca		ya
	ra		nya
	ka		ma
	da		ga
	ta		ba
	sa		tha
	wa		nga
	la		re
	pa		le

TABLE II
NGLEGENA CHARACTERS ALONG WITH THEIR PRONUNCIATION USING ROMAN CHARACTERS

Sandangan	Name	Pronounciation
	wulu	i
	suku	u
	taling	e
	pepet	é
	taling tarung	o

B. Preprocessing

1) Grayscaling: In Javanese script transliteration, grayscaling is needed because it can help to improve the accuracy of Optical Character Recognition (OCR) technology [13]. When scanned or photographed, a Javanese script is captured as an image with shades of black and white. This image can be difficult for OCR algorithms to recognize, especially if there are variations in the background, font type, or size [14]. Grayscaling the image can help simplify it by converting it into a grayscale format with a range of gray shades, from white to black [15], [16]. This process can help reduce the image's complexity and make it easier for the OCR algorithm to recognize the characters. Grayscaling can also help to enhance the contrast between the text and background, which can further improve the accuracy of the OCR algorithm [17].

a

b

Fig. 2 The Javanese script before (a) and after (b) shadows removal

2) Gaussian Blurring: In Javanese script transliteration, Gaussian blurring is used as a preprocessing step to smooth out the image and reduce noise before performing character recognition [18]. This is important because the script contains many fine details and curves, and any noise or irregularities in the image can cause misinterpretation or misclassification of the characters. Gaussian blurring helps to reduce these issues by smoothing out the image and removing any small variations in intensity that might be caused by noise or uneven lighting [19]. This results in a more transparent and uniform image, which can improve the accuracy of character recognition algorithms.

3) Shadow Removal: The dilation operation contributes to the shadow removal process indirectly. The dilation is a morphological operation initially applied to the input image using a large square kernel. This operation enlarges the boundaries of foreground objects and fills in small gaps, effectively thickening and enhancing certain features within the image [20]. Following the initial dilation, the function computes the background image by performing dilation again on the inverted dilated image. This process helps approximate the image's background, encompassing shadow regions [21]. While dilation is not directly employed for shadow removal, it is an essential component of the background extraction process. Extracting the background, subsequent operations, such as the difference image and thresholding computation, can effectively identify and isolate shadow regions for removal. Figure 2 shows an example of shadow removal.

4) Thresholding: In Javanese script transliteration, thresholding separates the foreground from the image's background and extracts the text regions of interest. The text regions can be easily identified and segmented from the background by converting the grayscale image to binary. This allows for more accurate recognition and transliteration of the Javanese script characters. Thresholding separates the foreground from the background of the image and extracts the text regions of interest. This process enhances the accuracy of Javanese script character recognition and transliteration by reducing noise and improving image contrast. Additionally, thresholding helps reduce noise and improve the image's contrast, further enhancing the transliteration process's accuracy. This study uses Otsu thresholding [22].

5) Character Improvement: In Javanese script transliteration, dilation thickens or expands the characters. This is because some characters in Javanese script can appear thin or faint due to the quality of the image or the writing style, which can cause difficulties in character recognition. By applying dilation, the characters can be made thicker and more visible, which can improve the accuracy of the character recognition process. Dilation can also fill in gaps or breaks in the characters, which can occur due to the poor quality of the writing or image, making it easier to recognize the character [23]. Overall, dilation can help improve character recognition accuracy in Javanese script transliteration.

C. Projection Profile Segmentation

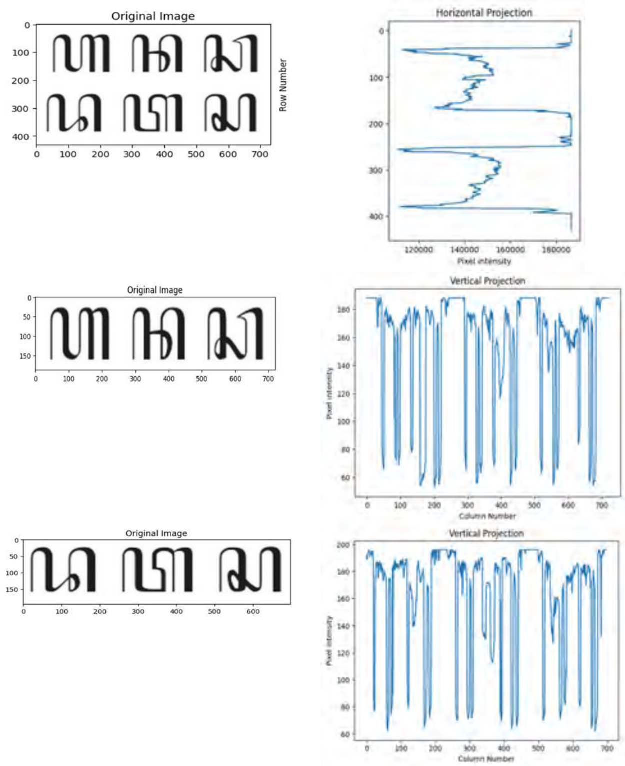
Projection Profile Segmentation is a method used in image processing to segment an image into multiple regions by analyzing the image's horizontal and vertical projections [24], [25]. In the case of Javanese script transliteration, Projection Profile Segmentation can be used to separate individual characters from a continuous line of text. The formula for the horizontal projection profile can be seen in Eq 1.

$$h(j) = \sum_{i=1}^{m} I(i,j) \quad (1)$$

where $h(j)$ is the sum of the intensity values of all pixels in the column j, $I(i,j)$ is the intensity value of the pixel at row i and column j, and m is the number of rows in the image. Similarly, the formula for the vertical projection profile can be seen in Eq 2.

$$v(i) = \sum_{j=1}^{n} I(i,j) \quad (2)$$

where $v(j)$ is the sum of the intensity values of all pixels in column j, $I(i,j)$ is the intensity value of the pixel at row i and column j, and n is the number of rows in the image. Figure 3 shows an example of horizontal and vertical projection in Javanese script.



a

b

Fig. 3 Example of horizontg.al and vertical projection in Javanese script. The original images (a) and their pixel intensity based on projection (b)

D. Classification

Following the segmentation process, this study progresses to character classification within Javanese script using transfer learning, which repurposes pre-trained models for similar tasks. This study employed three pre-trained models: MobileNetV2, VGG-16, and Xception, each offering distinct advantages in image analysis tasks.

MobileNetV2 is celebrated for its lightweight architecture and efficiency, making it particularly suited for deployment on mobile devices owing to its minimal computational demands [26]. This model achieves its efficiency through the innovative use of depthwise separable convolutions, which significantly reduce the number of parameters without sacrificing performance. MobileNetV2's streamlined architecture enables rapid inference and deployment, making it an attractive choice for resource-constrained environments.

In contrast, VGG-16 stands out for its depth and simplicity, comprising 16 layers with relatively small (3x3) convolutional filters [27]. Despite its straightforward architecture, VGG-16 has demonstrated remarkable performance in various image classification tasks. Its architecture's simplicity and uniformity contribute to ease of understanding and implementation, facilitating its widespread adoption in computer vision applications.

Xception represents a departure from conventional CNN by introducing a novel architecture based on depth-wise separable convolutions. Unlike traditional CNNs, Xception employs depthwise separable convolutions at every layer, facilitating the extraction of complex features while significantly reducing computational overhead [28]. This innovative architecture enables Xception to achieve state-of-the-art performance across various image recognition tasks. By leveraging these diverse models, our study aims to evaluate their performance and identify the most suitable candidate for Javanese to Roman script transliteration. Transfer learning allows us to adapt these pre-trained models to our specific task, leveraging their learned representations to enhance the accuracy of character classification within Javanese script. Through meticulous examination, rigorous evaluation, and comprehensive comparison across various performance metrics, this study endeavors to ascertain the model that provides the most favorable equilibrium among accuracy, efficiency, and scalability, explicitly tailored to the exigencies and complexities inherent in our transliteration task.

III. Results and Discussion

A. Character Classification Results

The first experiment conducted classification tests on single printed Javanese characters across three transfer learning architectures: MobileNetV2, VGG-16, and Xception. All models underwent training using an identical dataset, comprising training and testing subsets, obtained through the segmentation process during dataset collection. The evaluation encompassed 46 distinct classes, each containing 40 images. The evaluation methodology adopted the K-Fold technique with 5 folds, and each training epoch consisted of 15 iterations. The initial model, employing the MobileNetV2 architecture, yielded a compact model size of 9 MB. Notably, the average F1 score achieved perfection across all folds, denoted by a score of 1, indicative of impeccable classification performance in every fold. Subsequently, the VGG-16 architecture was employed, resulting in a larger model size of 512 MB. Despite a commendable average F1 score of 0.985 across all folds, indicative of high performance, discernible errors were observed during the classification process in the evaluation phase. Lastly, the Xception architecture, with a model size of 80.1 MB, achieved consistently flawless classification results, reflected in an average F1 score of 1 across all folds during the evaluation phase. Detailed F1 scores for each model's classification of individual classes for the first experiment are presented in Table 3.

TABLE IIII
CLASSIFICATION PERFORMANCE RESULTS IN THE FIRST EXPERIMENT

Model	F1 Score	Model Size (MB)
MobileNetV2	1	9
VGG-16	0.985	512
Exception	1	80.1

The second experiment involved creating models using a combined dataset comprising printed Javanese nglegena script, obtained from the preceding phase, and handwritten Javanese nglegena script by Phiard et al. [29], with 40 randomly selected images per class, resulting in a total dataset of 80 images per class. This trial encompassed 20 classes, specifically only nglegena characters. Similar to the system model construction, the architectures employed were MobileNetV2, VGG-16, and Xception. The model evaluation utilized the K-Fold cross-validation with 5 folds; each training process included 15 epochs. The first model, employing the MobileNetV2 architecture, achieved an evaluation score of 0.992 for F1 score. The second model, utilizing the Xception architecture, attained an evaluation score of 0.989 for F1 score. Conversely, the model employing the VGG-16 architecture garnered F1 score of 0.434. Compared to the models explicitly developed from the first experiment, there was a decrease in overall accuracy or evaluation scores for each model. This decline can be attributed to the variation of handwritten and printed scripts, which increased the dataset's diversity. Additionally, it implies that models created with mixed datasets exhibit higher error rates, particularly for the VGG-16 architecture. Detailed F1 scores for each model's classification of individual classes for the second experiment are presented in Table 4.

TABLE IV
CLASSIFICATION PERFORMANCE RESULTS IN THE SECOND EXPERIMENT

Model	F1 Score	Model Size (MB)
MobileNetV2	0.992	9
VGG-16	0.434	512
Exception	0.989	80.1

B. Transliteration Results

The method required in the transliteration process involves the overall segmentation and classification of segmentation results. This method leverages preprocessing methods that were established beforehand. The models used in the transliteration experiment are generated during the model training process: the MobileNetV, VGG-16, and Xception. To assess the transliteration results, a similarity score based on the Levenshtein Distance of the transliteration results for each image and each model was utilized, also normalized based on the length of the original text and the transliteration results using TheFuzz library [30]. The Levenshtein Distance was chosen due to its advantages in assessing the similarity or differences between two strings, owing to its speed and suitability for evaluating string similarity [31], [32]. Another advantage of this method is its ability to handle string comparisons with differing character counts, aligning with the classification results prone to classification errors by the model or spacing differences, which are also accounted for by this method [33].

Fig. 4 Example of Transliteration Results

This method successfully segmented and classified the Javanese script images based on the experimental outcomes, yielding varying average similarity scores for each model. Figure 4 shows an example of the transliteration process. Table 5 shows the similarity score from each architecture. The models demonstrated commendable capability in recognition or transliteration, akin to the preceding trials, with most trial images accurately classified. Transliteration errors primarily occurred during the segmentation stage. Images were susceptible to segmentation errors owing to the image capture angles, such as with characters "le" and "ga," or characters with the sandhangan wulu potentially not separated by a vertical line and instead interpreted as a single character.

Several images experienced transliteration errors due to system limitations in recognizing "lighttext" text types that may contain shadows. Additionally, characters with thin lines may be segmented into two distinct characters due to discontinuation during thresholding. Non-existent characters in images may emerge from segmentation due to noise on computer monitor components that are unremoved during blurring and shadow removal processes. Images with excessive zoom or small text size may also lead to transliteration errors due to high image noise sharpness.

TABLE V
SIMILARITY SCORE FOR EACH ARCHITECTURE

Model	Similarity Score (%)
MobileNetV2	90.07
VGG-16	89.26
Exception	90.78

In addition to accuracy, the time required for transliteration from image reading to producing Romanized results was recorded for each image in each model. Table 6.3 illustrates the time taken for the entire transliteration process of the trial images. Based on the overall transliteration process (excluding image upload time), the MobileNet V2 architecture model exhibited the fastest transliteration process, with an average transliteration time of 1.57 seconds for all trial images. The Xception architecture model ranked second fastest, with an average duration of 5.38 seconds. The VGG-16 architecture model required the longest time for transliteration, with an average transliteration duration of 7.26 seconds. Table 6 shows the transliteration time for each architecture.

TABLE VI
TRANSLITERATION TIME

Model	Time (s)
MobileNetV2	1.57
VGG-16	7.26
Exception	5.38

C. Comparison with Previous Studies

This study conducted a comprehensive comparison with previous studies in Javanese script transliteration. While most previous studies solely reported the accuracy of classification results, our research offers a more nuanced evaluation. This study introduced additional performance metrics, including F1 score, transliteration score using Levenshtein distance, and transliteration time. These metrics provide a more holistic assessment of the transliteration process, capturing both classification accuracy and the efficiency of transliteration. However, it's important to note that comparisons with previous studies may be limited due to differences in datasets and experimental setups. While this study and previous ones utilize private datasets, variations in data collection methods, dataset sizes, and preprocessing techniques may impact results. Therefore, while this study contributes additional insights into Javanese script transliteration performance, direct comparisons should be made cautiously, considering these potential discrepancies. Table 7 shows the performance comparison with previous studies.

TABLE VII
COMPARISON WITH PREVIOUS STUDIES

Research	Accuracy (%)	F1 Score (%)	Similarity Score (%)	Transliteration time (s)
Dewa et al [10]	89	-	-	-
Widiarti et al [11]	87.29	-	-	-
Ilham et al [12]	80	-	-	-
WIbowo et al [6]	96.21	-	-	-
Robby et al [4]	97.5	-	-	-
Proposed method (MobileNetv2)	99	99	90.07	1.57
Proposed method (VGG-16)	98.5	71	89.26	7.26
Proposed method (Exception)	99	99	90.78	5.38

Including the F1 score allows for a balanced evaluation of precision and recall, offering insights beyond simple accuracy measurements. Similarly, the transliteration score using

Levenshtein distance provides a quantitative measure of the similarity between the original Javanese script and the Romanized text, offering a more granular assessment of transliteration accuracy. Additionally, considering transliteration time provides valuable information on the computational efficiency of the models, essential for real-time applications.

D. Discussion

The results of the experiments shed light on the efficacy of different transfer learning architectures in Javanese script transliteration. The first experiment, focusing on single printed Javanese characters, demonstrated remarkable classification performance across all three models: MobileNetV2, VGG-16, and Xception. The MobileNetV2 architecture, known for its lightweight and efficient design, exhibited flawless classification results, suggesting its suitability for tasks with computational constraints. Conversely, while the VGG-16 architecture achieved a high F1 score, it showcased discernible errors during the evaluation phase, possibly due to its deeper architecture and higher computational demands. The Xception architecture, distinguished for its feature extraction capabilities, consistently yielded flawless classification results, underscoring its effectiveness in complex classification tasks.

In the second experiment, where models were trained on a combined dataset of printed and handwritten Javanese nglegena script, a decline in overall accuracy was observed compared to models trained solely on printed script. This decline underscores the challenges posed by dataset diversity, particularly evident in the VGG-16 architecture model, which exhibited a significant decrease in performance. The combination of handwritten and printed scripts introduces complexities in classification, leading to higher error rates, especially in architectures that could be more adept at handling dataset variations.

The transliteration process, utilizing the Levenshtein Distance metric for similarity scoring, revealed the effectiveness of the MobileNetV2 and Xception architectures in accurately transcribing Javanese script to Romanized text. However, segmentation errors were identified as a primary source of transliteration inaccuracies, particularly in characters with intricate details or shadows. System limitations in recognizing "light text" and noise artifacts further contributed to transliteration errors, emphasizing the need for robust preprocessing techniques. Moreover, the transliteration time analysis highlighted the computational efficiency of the MobileNetV2 architecture, which exhibited the shortest transliteration duration compared to VGG-16 and Xception. This efficiency underscores its suitability for real-time applications where swift processing is paramount.

The obtained results from the experiments provide valuable insights into the performance of various transfer learning architectures in Javanese script transliteration, offering implications across multiple dimensions. Firstly, the F1 score serves as a critical metric for evaluating the classification performance of the transfer learning models. While all three architectures—MobileNetV2, VGG-16, and Xception—demonstrated high F1 scores, the discernible errors observed in the VGG-16 architecture during the evaluation phase suggest potential challenges associated with its deeper architecture and higher computational demands. Secondly, analyzing the similarity score derived from the transliteration process provides crucial insights into the accuracy of transcribing Javanese script into Romanized text. While both MobileNetV2 and Xception architectures demonstrated commendable accuracy, identifying segmentation errors as a primary source of inaccuracies underscores the significance of robust preprocessing techniques. These errors, compounded by system limitations in recognizing certain text types and noise artifacts, highlight the need for further refinement in preprocessing methodologies to enhance transliteration accuracy. Lastly, the analysis of transliteration time underscores the importance of computational efficiency, particularly in real-time applications where swift processing is essential. The MobileNetV2 architecture exhibited the shortest transliteration duration, indicating its suitability for scenarios where rapid processing is paramount. Conversely, the longer transliteration durations observed with the VGG-16 and Xception architectures emphasize the trade-off between computational complexity and processing time.

The results of the experiments shed light on the efficacy of different transfer learning architectures in Javanese script transliteration, highlighting their respective advantages and disadvantages. MobileNetV2, known for its lightweight and efficient design, exhibited flawless classification results and the shortest transliteration duration, making it ideal for tasks with computational constraints and real-time applications. Conversely, VGG-16 achieved a high F1 score but showed discernible errors during evaluation, likely due to its deeper architecture and higher computational demands, leading to longer transliteration times. Xception, distinguished for its powerful feature extraction capabilities, consistently yielded flawless classification results and high transliteration accuracy, but at the cost of a longer transliteration duration similar to VGG-16. The decline in overall accuracy observed in the second experiment, where models were trained on a combined dataset of printed and handwritten Javanese nglegena script, underscores the challenges of dataset diversity. VGG-16 exhibited a significant performance decrease with the combined dataset, highlighting the complexities in classification introduced by handwritten characters and the need for robust preprocessing techniques. Despite these challenges, the detailed analyses using metrics such as F1 score, Levenshtein distance for similarity scoring, and transliteration time provide comprehensive insights into the performance of these models, emphasizing the trade-offs between computational complexity, processing time, and accuracy.

The decline in overall accuracy observed in the second experiment, where models were trained on a combined dataset of printed and handwritten Javanese nglegena script, underscores the challenges of dataset diversity—the combination of handwritten and printed scripts introduced complexities in classification, leading to higher error rates. Handwritten characters often exhibit variations in style, stroke thickness, and imperfections that are absent in printed characters, making them harder to classify. This variability can confuse models primarily trained on more uniform printed characters.

IV. Conclusion

The experiments systematically evaluate various transfer learning architectures for Javanese script transliteration. MobileNetV2, VGG-16, and Xception demonstrated notable classification performance in the first experiment, with MobileNetV2 standing out for its lightweight design, while VGG-16 faced challenges likely due to its deeper architecture. The second experiment, incorporating both printed and handwritten scripts, revealed a decline in the F1 Score, particularly in VGG-16. Transliteration inaccuracies, stemming from segmentation errors and system limitations, underscored the importance of robust preprocessing. Notably, MobileNetV2 and Xception excelled in transliteration, with MobileNetV2 exhibiting the shortest processing time, rendering it apt for real-time applications.

In specific achievements, MobileNetV2 emerged as the optimal architecture for Javanese character classification, boasting a remarkable 99.2% F1 score. On the other hand, Xception proved most effective for Javanese script transliteration, achieving a commendable similarity score of 90.78%. The suitability of MobileNetV2 is further emphasized by its lightweight architecture, making it particularly conducive for mobile devices.

The study underscores the significance of architectural considerations and dataset characteristics in crafting proficient transliteration models for Javanese script. Future investigations could delve into advanced preprocessing techniques and architecture-specific optimizations to address segmentation errors, ultimately enhancing overall transliteration accuracy and efficiency.

The research findings hold significant practical implications for both academic and real-world applications in Javanese script transliteration. By demonstrating the effectiveness of MobileNetV2 for character classification and Xception for script transliteration, the study paves the way for developing user-friendly tools that facilitate easier Javanese script transcription. These findings advance computational linguistics research in academic contexts, offering insights into optimal model selection for script recognition and transliteration tasks. Moreover, in practical applications, such as mobile devices or digital platforms, the identified architectures can streamline the Javanese script conversion into Romanized text, promoting cultural heritage preservation and linguistic accessibility. Further research endeavors could refine transliteration models to address segmentation errors and enhance overall efficiency, supporting broader adoption and utilization in academic and practical domains.

References

[1] S. Wilonoyudho, “Urbanization and Regional Imbalances in Indonesia,” *Indones. J. Geogr.*, vol. 49, no. 2, p. 125, Dec. 2017, doi:10.22146/ijg.13039.

[2] A. R. Widiarti and R. Pulungan, “A method for solving scriptio continua in Javanese manuscript transliteration,” *Heliyon*, vol. 6, no. 4, p. e03827, Apr. 2020, doi: 10.1016/j.heliyon.2020.e03827.

[3] I. Prihandi, I. Ranggadara, S. Dwiasnati, Y. S. Sari, and Suhendra, “Implementation of Backpropagation Method for Identified Javanese Scripts,” *J. Phys. Conf. Ser.*, vol. 1477, no. 3, p. 032020, Mar. 2020, doi: 10.1088/1742-6596/1477/3/032020.

[4] G. A. 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 Comput. Sci.*, vol. 157, pp. 499–505, 2019, doi:10.1016/j.procs.2019.09.006.

[5] A. Susanto, C. Atika Sari, I. U. W. Mulyono, and M. Doheir, “Histogram of Gradient in K-Nearest Neighbor for Javanese Alphabet Classification,” *Sci. J. Inform.*, vol. 8, no. 2, pp. 289–296, Nov. 2021, doi: 10.15294/sji.v8i2.30788.

[6] G. H. Wibowo, R. Sigit, and A. Barakbah, “Javanese Character Feature Extraction Based on Shape Energy,” *Emit. Int. J. Eng. Technol.*, vol. 5, no. 1, pp. 154–169, Jul. 2017, doi:10.24003/emitter.v5i1.175.

[7] B. E. Praheto and F. B. B. Utomo, “Transliteration Method In Learning Reading Of The Javanese Script,” 2019.

[8] Y. Harjoseputro, Y. D. Handarkho, and H. T. R. Adie, “The Javanese Letters Classifier with Mobile Client-Server Architecture and Convolution Neural Network Method,” *Int. J. Interact. Mob. Technol. IJIM*, vol. 13, no. 12, p. 67, Dec. 2019, doi: 10.3991/ijim.v13i12.11492.

[9] A. Setiawan, A. S. Prabowo, and E. Y. Puspaningrum, “Handwriting Character Recognition Javanese Letters Based on Artificial Neural Network,” vol. 1, no. 1, 2019.

[10] C. K. Dewa, A. L. Fadhilah, and A. Afiahayati, “Convolutional Neural Networks for Handwritten Javanese Character Recognition,” *IJCCS Indones. J. Comput. Cybern. Syst.*, vol. 12, no. 1, p. 83, Jan. 2018, doi:10.22146/ijccs.31144.

[11] A. R. Widiarti, R. Pulungan, A. Harjoko, Marsono, and S. Hartati, “A proposed model for Javanese manuscript images transliteration,” *J. Phys. Conf. Ser.*, vol. 1098, p. 012014, Sep. 2018, doi: 10.1088/1742-6596/1098/1/012014.

[12] F. Ilham and N. Rochmawati, “Transliterasi Aksara Jawa Tulisan Tangan ke Tulisan Latin Menggunakan CNN,” *J. Inform. Comput. Sci. JINACS*, vol. 1, no. 04, pp. 200–208, Jul. 2020, doi:10.26740/jinacs.v1n04.p200-208.

[13] A. P. Tafti, A. Baghaie, M. Assefi, H. R. Arabnia, Z. Yu, and P. Peissig, “OCR as a Service: An Experimental Evaluation of Google Docs OCR, Tesseract, ABBYY FineReader, and Transym,” Advances in Visual Computing, pp. 735–746, 2016, doi: 10.1007/978-3-319-50835-1_66.

[14] J. Tse, C. Jones, D. Curtis, and E. Yfantis, “An OCR-independent character segmentation using shortest-path in grayscale document images,” in *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, Cincinnati, OH, USA: IEEE, Dec. 2007, pp. 142–147. doi: 10.1109/ICMLA.2007.21.

[15] I. Zeger, S. Grgic, J. Vukovic, and G. Sisul, “Grayscale Image Colorization Methods: Overview and Evaluation,” *IEEE Access*, vol. 9, pp. 113326–113346, 2021, doi: 10.1109/ACCESS.2021.3104515.

[16] S. Wan, Y. Xia, L. Qi, Y.-H. Yang, and M. Atiquzzaman, “Automated Colorization of a Grayscale Image With Seed Points Propagation,” *IEEE Trans. Multimed.*, vol. 22, no. 7, pp. 1756–1768, Jul. 2020, doi:10.1109/TMM.2020.2976573.

[17] M. Hagara, R. Stojanović, T. Bagala, P. Kubinec, and O. Ondráček, “Grayscale image formats for edge detection and for its FPGA implementation,” *Microprocess. Microsyst.*, vol. 75, p. 103056, Jun. 2020, doi: 10.1016/j.micpro.2020.103056.

[18] H. Guo, D. Yang, J. Zhao, and Y. Liu, “Research on the Preprocessing of Tai Le Character Recognition,” *J. Phys. Conf. Ser.*, vol. 1617, no. 1, p. 012039, Aug. 2020, doi: 10.1088/1742-6596/1617/1/012039.

[19] Y.-Q. Liu, X. Du, H.-L. Shen, and S.-J. Chen, “Estimating Generalized Gaussian Blur Kernels for Out-of-Focus Image Deblurring,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 3, pp. 829–843, Mar. 2021, doi: 10.1109/TCSVT.2020.2990623.

[20] K. A. M. Said and A. B. Jambek, “Analysis of Image Processing Using Morphological Erosion and Dilation,” *J. Phys. Conf. Ser.*, vol. 2071, no. 1, p. 012033, Oct. 2021, doi: 10.1088/1742-6596/2071/1/012033.

[21] B. Amin, M. Mohsin Riaz, and A. Ghafoor, “Automatic shadow detection and removal using image matting,” *Signal Process.*, vol. 170, p. 107415, May 2020, doi: 10.1016/j.sigpro.2019.107415.

[22] M. T. Nyo, F. Mebarek-Oudina, S. S. Hlaing, and N. A. Khan, “Otsu's thresholding technique for MRI image brain tumor segmentation,” *Multimed. Tools Appl.*, vol. 81, no. 30, pp. 43837–43849, Dec. 2022, doi: 10.1007/s11042-022-13215-1.

[23] M. Khayyat, L. Lam, C. Y. Suen, F. Yin, and C.-L. Liu, “Arabic Handwritten Text Line Extraction by Applying an Adaptive Mask to Morphological Dilation,” in *2012 10th IAPR International Workshop on Document Analysis Systems*, Gold Coast, Queenslands, TBD, Australia: IEEE, Mar. 2012, pp. 100–104. doi: 10.1109/DAS.2012.20.

[24] R. Ptak, B. Żygadło, and O. Unold, “Projection–Based Text Line Segmentation with a Variable Threshold,” *Int. J. Appl. Math. Comput. Sci.*, vol. 27, no. 1, pp. 195–206, Mar. 2017, doi: 10.1515/amcs-2017-0014.

[25] K. L. Banumathi and A. P. Jagadeesh Chandra, “Line and word segmentation of Kannada handwritten text documents using projection profile technique,” in *2016 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT)*, Mysuru: IEEE, Dec. 2016, pp. 196–201. doi:10.1109/ICEECCOT.2016.7955214.

[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT: IEEE, Jun. 2018, pp. 4510–4520. doi:10.1109/CVPR.2018.00474.

[27] S. Mascarenhas and M. Agarwal, “A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification,” in *2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, Bengaluru, India: IEEE, Nov. 2021, pp. 96–99. doi:10.1109/centcon52345.2021.9687944.

[28] F. Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI: IEEE, Jul. 2017, pp. 1800–1807. doi: 10.1109/CVPR.2017.195.

[29] Phiard, “Aksara Jawa.” 2020. [Online]. Available: https://www.kaggle.com/datasets/phiard/aksara-jawa/data

[30] Setgeek, “Thefuzz.” 2022. [online] Available: https://github.com/seatgeek/thefuzz.

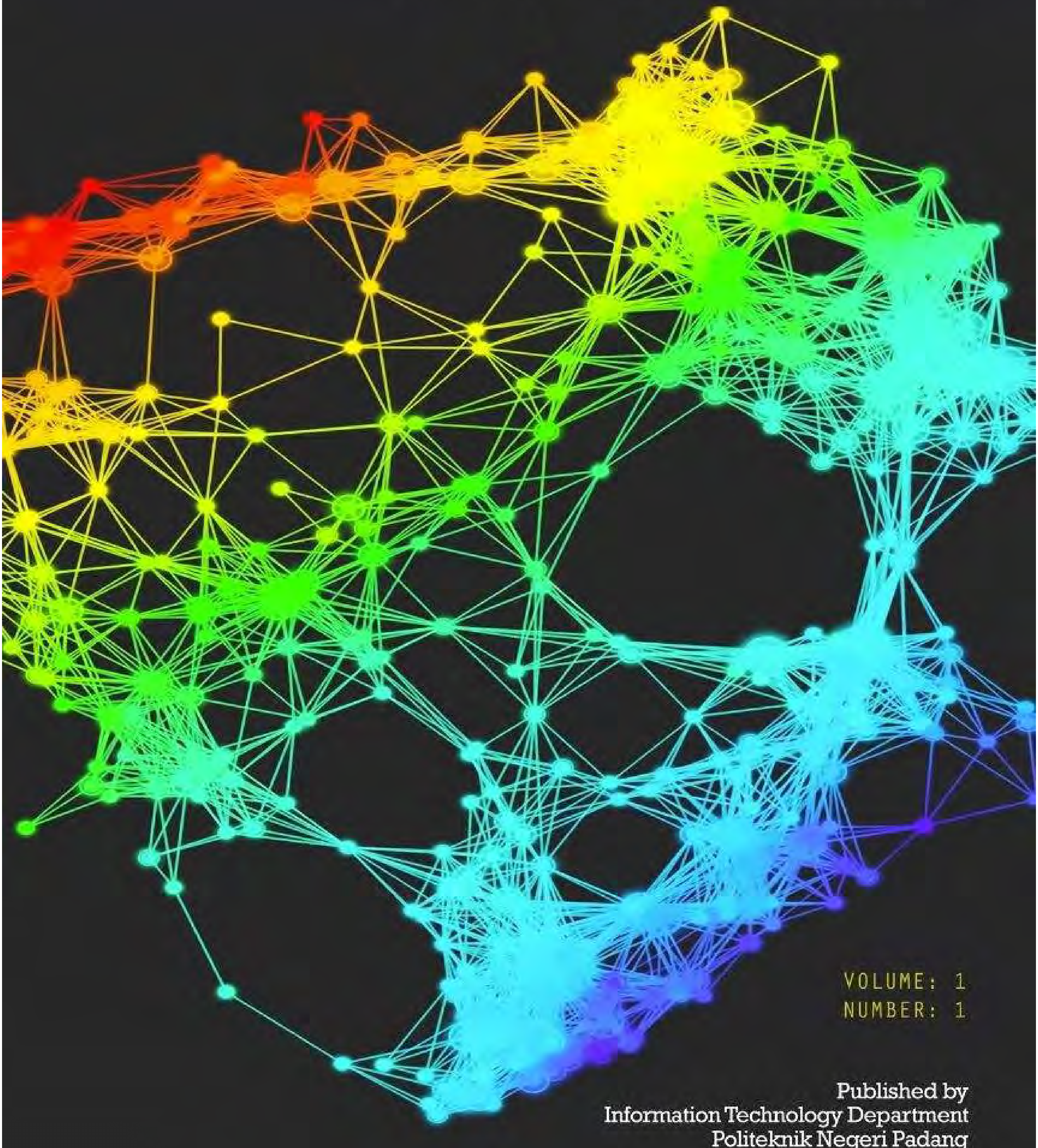
[31] D. K. Po, “Similarity Based Information Retrieval Using Levenshtein Distance Algorithm,” *Int. J. Adv. Sci. Res. Eng.*, vol. 06, no. 04, pp. 06–10, 2020, doi: 10.31695/IJASRE.2020.33780.

[32] Sugiarto, I. G. S. M. Diyasa, and I. N. Diana, “Levenshtein Distance Algorithm Analysis on Enrollment and Disposition of Letters Application,” in *2020 6th Information Technology International Seminar (ITIS)*, Surabaya, Indonesia: IEEE, Oct. 2020, pp. 198–202. doi: 10.1109/ITIS50118.2020.9321030.

[33] M. H. Al Farisi *et al.*, “K-Means Algorithm and Levenshtein Distance Algorithm for Sentiment Analysis of School Zonation System Policy,” in *2021 Sixth International Conference on Informatics and Computing (ICIC)*, Jakarta, Indonesia: IEEE, Nov. 2021, pp. 1–6. doi:10.1109/ICIC54025.2021.9632943.

JOIV

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION



VOLUME: 1
NUMBER: 1

Published by
Information Technology Department
Politeknik Negeri Padang

ISSN 2549-9610 (PRINT)
ISSN 2549-9904 (ONLINE)

JOIV INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

Editorial Team

QUICK MENU

- Editorial Team
- Focus & Scope
- Indexing
- Author Guidelines
- Peer Review Process
- Author Fees
- Publication Ethics
- Online Submission
- Open Access Statement
- Plagiarism Policy
- Special Issuess
- Licensing terms
- Contact

REQUEST INDEXING

- **SCOPUS (ACCEPTED)**
  - Submission Received: **March 3, 2020**
  - Submission Accepted: **July 30, 2020**
  - **SCOPUS CiteScore Tracker 2020**
- **WoS / Web of Science**
  - Latest submission: September 16, 2018
  - **Web of Science** Citation Analysis
- **IET INSPEC**
  - Added to review: **May 29, 2020**

ISSN 2549-9610 (PRINT)
ISSN 2549-9904 (ONLINE)
JOIV INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

Vol 8, No 3 (2024)

This issue has been available online since **30th September 2024** for the regular issue of September 2024. All articles in this issue (**50 original research articles**) were authored/co-authored by **216 authors** from **13 countries (Finland, Indonesia, India, Iraq, Jordan, Japan, Malaysia, Morocco, Nigeria, Philippines, Saudi Arabia, South Africa** and **United Kingdom).**

Table of Contents

Regular

Issue Coverage
(216 Authors)

Total 13 Author's Countries

Finland	(1)
United Kingdom	(2)
Indonesia	(127)
India	(1)
Iraq	(11)
Jordan	(2)
Japan	(3)
Morocco	(2)
Malaysia	(63)
Nigeria	(1)
Philippines	(1)
Saudi Arabia	(1)
South Africa	(1)

www.joiv.org

REQUEST INDEXING

- **SCOPUS (ACCEPTED)**
  - Submission Received: **March 3, 2020**
  - Submission Accepted: **July 30, 2020**
  - **SCOPUS CiteScore Tracker 2020**
- **WoS / Web of Science**
  - Latest submission: September 16, 2018
  - **Web of Science** Citation Analysis
- **IET INSPEC**
  - Added to review: **May 29, 2020**
- **Ei COMPENDEX**
  - Submission: **February 10, 2021**

PUBLICATION PARTNERS

INSTITUTE OF VISUAL INFORMATICS

Soft Computing and Data Mining Centre

USER

Username
Password
Remember me
Login

JOIV : International Journal on Informatics Visualization
ISSN 2549-9610 **(print)** | 2549-9904 **(online)**
Organized by Society of Visual Informatics, and Institute of Visual Informatics - UKM and Soft Computing and Data Mining Centre - UTHM
W : http://joiv.org
E : alde@sotvi.org , rahmat@sotvi.org
View JOIV Stats

International Journal on Informatics Visualization

COUNTRY

Indonesia

Universities and research institutions in Indonesia

Media Ranking in Indonesia

SUBJECT AREA AND CATEGORY

Computer Science
- Computer Science (miscellaneous)

Decision Sciences
- Information Systems and Management
- Statistics, Probability and Uncertainty

PUBLISHER

Politeknik Negeri Padang

H-INDEX

15

PUBLICATION TYPE

Journals

ISSN

25499904

COVERAGE

2017-2023

INFORMATION

Homepage

How to publish in this journal

joiv@pnp.ac.id

SCOPE

The journal scopes include (but not limited to) the followings: Computer Science : Artificial Intelligence, Data Mining, Database, Data Warehouse, Big Data, Machine Learning, Operating System, Algorithm. Computer Engineering : Computer Architecture, Computer Network, Computer Security, Embedded system, Coud Computing, Internet of Thing, Robotics, Computer Hardware. Information Technology : Information System, Internet & Mobile Computing, Geographical Information System. Visualization : Virtual Reality, Augmented Reality, Multimedia, Computer Vision, Computer Graphics, Pattern & Speech Recognition, image processing. Social Informatics: ICT interaction with society, ICT application in social science, ICT as a social research tool, ICT education.

Join the conversation about this journal

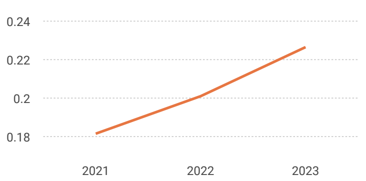
Quartiles

FIND SIMILAR JOURNALS

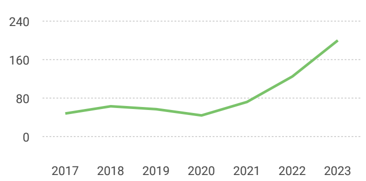
options

1 Journal of Theoretical and Applied Information PAK 74% similarity	2 International Journal of Advanced Computer Science GBR 73% similarity	3 Journal of Computer Science USA 70% similarity	4 International Journal of Computing and Digital BHR 70% similarity	5 Journal of Advances in Information Technology USA 69% similarity

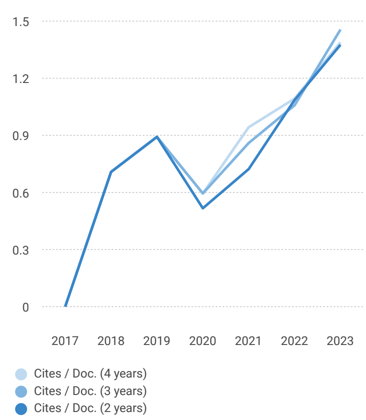
SJR



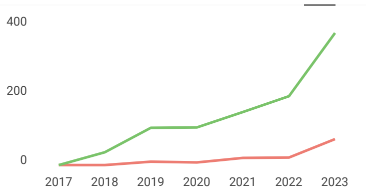
Total Documents



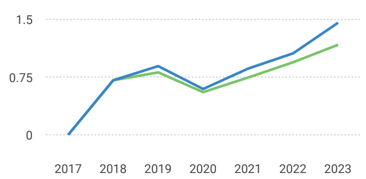
Citations per document



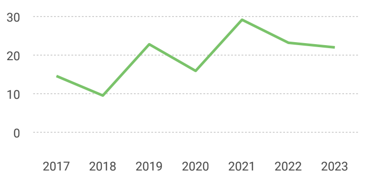
Total Cites Self-Cites



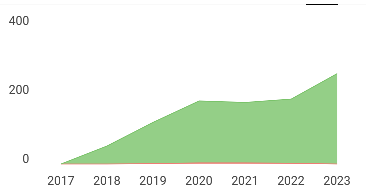
External Cites per Doc Cites per Doc



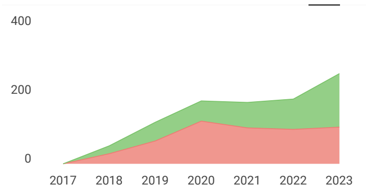
% International Collaboration



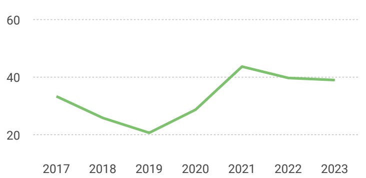
Citable documents Non-citable documents



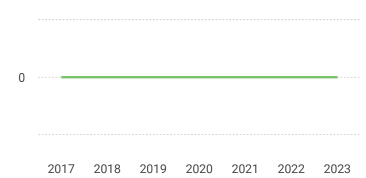
Cited documents Uncited documents



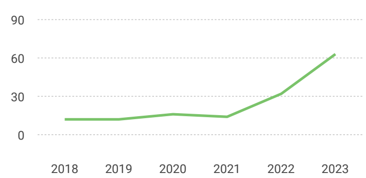
% Female Authors



Documents cited by public policy (Overton)



Documents related to SDGs (UN)



← Show this widget in your own website

Just copy the code below and paste within your html code:

<a href="https://www.scim

SCImago Graphica

Explore, visually communicate and make sense of data with our **new data visualization tool**.

Metrics based on Scopus® data as of March 2024

H **Hakeem flayyih** 10 months ago

The editor is not honest in dealing with researchers. I sent an article to the journal, and after three months of waiting, he claimed not to have received the article. After that, he told me that the article is suitable for the journal's scope. Until now, I have not received a response regarding the revisions or acceptance of the research. I am confused whether to send it to another journal or wait for the editor who excuses himself by claiming to be traveling between countries, sometimes in Kuwait and sometimes in New Zealand!

reply

Leave a comment

Name

Email
(will not be published)

Submit

The users of Scimago Journal & Country Rank have the possibility to dialogue through comments linked to a specific journal. The purpose is to have a forum in which general doubts about the processes of publication in the journal, experiences and other issues derived from the publication of papers are resolved. For topics on particular articles, maintain the dialogue through the usual channels with your editor.

Scopus Preview

Source details

International Journal on Informatics Visualization

Open Access

Years currently covered by Scopus: from 2017 to 2024

Publisher: Politeknik Negeri Padang

E-ISSN: 2549-9904

Subject area: Decision Sciences: Statistics, Probability and Uncertainty | Decision Sciences: Information Systems and Management | Computer Science: General Computer Science

Source type: Journal

View all documents > Set document alert Save to source list

CiteScore 2023
1.4

SJR 2023
0.211

SNIP 2023
0.403

CiteScore CiteScore rank & trend Scopus content coverage

CiteScore 2023

$$1.4 = \frac{\text{630 Citations 2020 - 2023}}{\text{441 Documents 2020 - 2023}}$$

Calculated on 05 May, 2024

CiteScoreTracker 2024

$$1.7 = \frac{\text{891 Citations to date}}{\text{523 Documents to date}}$$

Last updated on 05 October, 2024 • Updated monthly

CiteScore rank 2023

Category	Rank	Percentile
Decision Sciences — Statistics, Probability and Uncertainty	#114/168	32nd
Decision Sciences — Information Systems and Management	#106/148	28th

View CiteScore methodology > CiteScore FAQ > Add CiteScore to your site