JOIN (Jurnal Online Informatika)
p-ISSN: 2528-1682, e-ISSN: 2527-9165
Volume 8 Number 2 | December 2023: 232-241
DOI: 10.15575/join.v8i2.1180

YOLOv5 and U-Net-based Character Detection for Nusantara Script

Agi Prasetiadi[1], Julian Saputra[2], Iqsyahiro Kresna A[3], Imada Ramadhanti[4]
[1,2,3,4]Faculty of Informatics, Institut Teknologi Telkom Purwokerto, Indonesia

Article Info

Article history:

Received September 05 2023
Revised November 09, 2023
Accepted November 12, 2023
Published December 28, 2023

Keywords:

Nusantara Script
Character Detection
Bounding Box
YOLO
U-Net

ABSTRACT

Indonesia boasts a diverse range of indigenous scripts, called Nusantara scripts, which encompass Bali, Batak, Bugis, Javanese, Kawi, Kerinci, Lampung, Pallava, Rejang, and Sundanese scripts. However, prevailing character detection techniques predominantly cater to Latin or Chinese scripts. In an extension of our prior work, which concentrated on the classification of script types and character recognition within Nusantara script systems, this study advances our research by integrating object detection techniques, employing the YOLOv5 model, and enhancing performance through the incorporation of the U-Net model to facilitate the pinpointing of fundamental Nusantara script's character locations within input document images. Subsequently, our investigation delves into rearranging these character positions in alignment with the distinctive styles of Nusantara scripts. Experimental results reveal YOLOv5's performance, yielding a loss rate of approximately 0.05 in character location detection. Concurrently, the U-Net model exhibits an accuracy ranging from 75% to 90% for predicting character regions. While YOLOv5 may not achieve flawless detection of all Nusantara scripts, integrating the U-Net model significantly enhances the detection rate by 1.2%.

Corresponding Author:

Agi Prasetiadi,
Informatics Engineering, Faculty of Informatics, Institut Teknologi Telkom Purwokerto
Jl. D. I. Panjaitan No. 128, Purwokerto, Indonesia. 53147
Email: agi@ittelkom-pwt.ac.id

1. INTRODUCTION

There is no comprehensive Optical Character Recognition (OCR) deep learning model for detecting Nusantara scripts. As per Google Scholar data, research papers related to recognition and detection of Nusantara scripts, including Javanese, Batak, Balinese, Pallawa, Lontara, Kawi, Rejang, Sunda, Lampung, and Kerinci, vary in quantity, with counts in the 800s, 60s, 400s, 30s, 90s, 70s, 10s, 150s, 100s, and 10s respectively. Notably, Javanese, Balinese, and Sunda have received substantial recognition, while Pallawa and Rejang remain significantly underrepresented, lacking any implemented recognition or detection models. Additionally, among these studies, OCR topics receive attention primarily for Javanese and Balinese scripts, with approximately 10 research papers available, offering only partial detection capabilities.

OCR involves recognizing text in documents or images and converting it into machine-readable text [1]. It facilitates digitizing document content and improving operational efficiency [2]. Businesses extensively use OCR to extract information from printed media sources [3]. However, Nusantara scripts, which encompass a collection of traditional scripts indigenous to Indonesia, have faced near-extinction due to contemporary developments [4]. To counteract this decline and safeguard these invaluable scripts, we conducted research with the aim of developing a program capable of reading Nusantara script-—a specialized OCR program for Nusantara script. This study employed a dataset comprising 10 script types.

In essence, there are several image classification methods to read characters, including Structured Segment Matching Method [5], Morphological Analysis [6], Discriminant based evaluator on extracted features on normalized binary image [7], Neural Network [8], and Depthwise Separable Convolutional Neural Network [9]. However, a fundamental phase in character location detection involves the application of the bounding box technique, which is employed to isolate and subsequently process individual characters for classification. Previous studies have explored a variety of algorithms for this purpose, including Convolutional Neural Networks (CNN), Long-Short Term Memory (LSTM), You Only Look Once (YOLO). The intricacies of each algorithm will be elaborated as follows.

YOLOv4 has been used to train a CNN model for recognizing printed Arabic characters, achieving an accuracy of 82.4% [10]. Other researchers explored methods for developing an OCR system for Sanskrit Manuscripts, using conventional feature extraction, heuristic methods, and machine learning approaches such as CNN, LSTM, or Bidirectional LSTM [11]. An OCR system for Javanese Script was created using Projection Profile Segmentation and Nearest Centroid Classifier, achieving a 93.88% success rate for line segmentation and 73.59% success rate for character segmentation, with a classification accuracy of 60.6% [12]. The Faster Region-based Convolutional Neural Network (Faster R-CNN) has also been employed in Javanese script detection, achieving accuracy ranging from 41.67% to 96.31% [13]. YOLOv4 has demonstrated an impressive detection rate of nearly 99.55% for Balinese scripts on traditional lontar manuscripts [14]. Despite various versions of YOLO, YOLOv5 outperforms other versions in challenging environments like underwater scenarios [15]. Finally, beyond character location detection, the Multi-stage attentional U-Net has proven effective in pinpointing text regions on documents such as invoices, receipts, and bank transfers [16].

In our case, to establish a comprehensive OCR system for Nusantara scripts, it is crucial to construct three distinct models: a script's type classification model, a character recognition model tailored to each Nusantara script, and a model for character location detection in Nusantara scripts. The first two models have been developed in previous research, as documented in [17]. This study, however, centers on the character detection model. The approach employed in this paper primarily focuses on basic character detection to facilitate a feasible preprocessing and training procedure. The models designated for dataset training consist of a fusion between the YOLOv5 and U-Net models. YOLOv5 serves as the foundation for Nusantara script detection due to its robustness, while U-Net helps by pinpointing areas of interest using masks, complementing YOLOv5 in case of character misdetection. Performance assessments of these models will be conducted based on their respective loss and accuracy metrics.

2. METHOD

2.1. Nusantara Script

Indonesia has a rich history of Nusantara scripts, which have been used since ancient times for language expression [18]. The Pallava script, influenced by Sanskrit, was the earliest in Indonesia and impacted regions like Sumatra, Java, Bugis, and Bali. Various Nusantara scripts, including Javanese, Balinese, Lontara (Bugis), Rejang, Lampung, Batak, Pallava, Kawi, Sundanese, and Kerinci scripts, have cultural significance but limited practical use in daily life [19]. Most Nusantara scripts belong to the Abugida type, where consonants and vowels are combined as the fundamental script [20]. This script behavior is called complex text rendering in the context of computing.

Take the Javanese letter 'ha' as an example. When combined with Swara diacritics, it can produce at least 5 variations. These can change by 4 variations when combined with Panyigeg diacritics and yet another 4 with Wyanjana. This results in 150 variations for a single 'ha' letter (1 × 6 × 5 × 5). Extrapolating this complexity to all basic letters in Javanese script leads to thousands or even tens of thousands of script variations. Dealing with this large number, we have chosen an approach that works with the basic letter before any modifications. This approach aims for faster, more efficient, and more accurate letter recognition.

2.2. Object Detection

Object detection algorithms are used in character detection to classify images into discrete objects [21]. Bounding box segmentation assigns an imaginary rectangular box to each identified object, dynamically sized to fit the specific object. It is defined by a set of coordinates, which may encompass (x1, y1) and (x2, y2), or a single coordinate (x1, y1) along with width (w) and height (h) parameters [22]. This research focuses on character location detection, as there are no readily available libraries for detecting Nusantara characters. Bounding Box is used to pinpoint character locations within images, as shown in Figure 1.

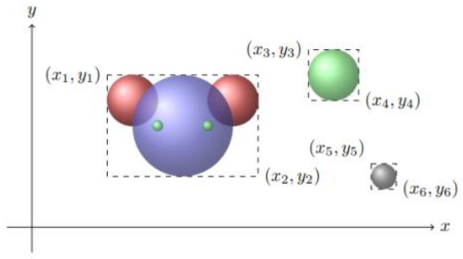


Figure 1. Object locations detected using coordinates x1, y1, x2, y2.

Figure 2. Example of object detection's application for latin text detection.

This technique utilizes a 4-dimensional matrix input for image analysis, resulting in output consisting of two coordinate pairs representing an object's location. Various architectures can be employed, including quadruplet CNN [23], Single Shot Detector (SSD) [24], YOLO [25], and Mask RCNN [26]. Figure 2 demonstrates object detection applied to text detection. Our research combines two key models: YOLOv5 and the U-Net model for character localization in Nusantara scripts.

YOLOv5, a CNN model for object detection [27] [28], stands out due to its impressive speed, achieving an FPS of 62.5, five times faster than alternatives like YOLOv3 and YOLOv4. Figure 3 presents YOLOv5's architecture, consisting of three main components: the Backbone, the Neck, and the Head. The Backbone extracts crucial image features for efficient data processing, employing the BottleNeckCSP (Bottleneck Cross-Stage Partial Networks) technique to reduce channel quantities while preserving spatial information [29].

The Neck refines features, reducing noise and improving information flow through structures like SPFF (Spatial Pyramid Pooling Fusion) and CSP-PAN (Cross Stage Partial-Path Aggregation Network). SPFF aggregates features from grid cells via MaxPooling, while CSP-PAN fosters information exchange across Backbone stages and captures features of varying abstraction levels. The final Head component refines outputs by NMS (Non-Maximum Suppression), eliminating duplicate bounding box predictions and ensuring the highest-confidence bounding box for each detected object [30].

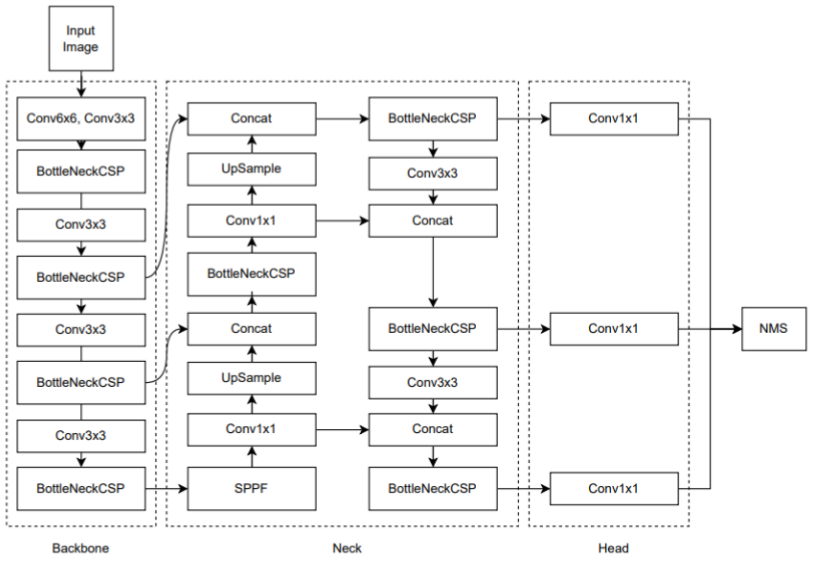


Figure 3. YOLOv5 model architecture.

U-Net is a deep learning model designed for biomedical image segmentation but has since been used for object segmentation tasks [31]. Figure 4 illustrates the architectural design of the U-Net model employed in this study. In this study, the U-Net model is used to predict the location of a text within an image, despite the presence of a banana such as shown in Figures 5a and 5b. The model uses five stages of downsampling and two stages of upsampling to convert an (128, 128) input image into a (32, 32) object map. The U-Net feature is then used to detect the locations of all Nusantara characters, enhancing the accuracy of the YOLOv5 model in character detection.

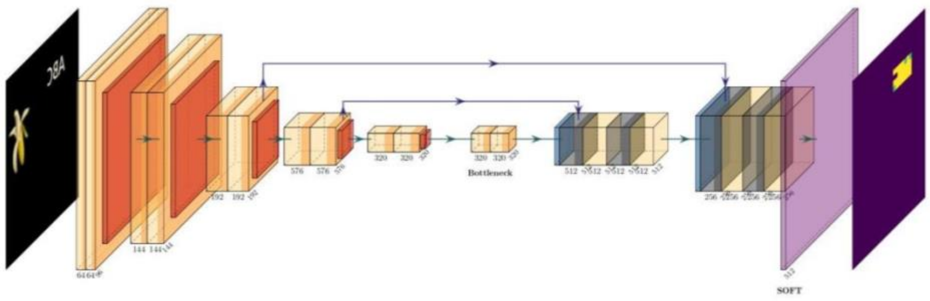


Figure 4. U-Net model architecture.

(a) Before detection. (b) After detection with red mask.

Figure 5. U-Net object mask detection for detecting a dog.

Because of YOLOv5's inherent nature, it does not always guarantee that all characters will be detected in a single pass. We employ an additional strategy of seeking potential character regions using the U-Net model to enhance the probability of capturing the remaining characters. As illustrated in Figure 6, the primary focus of character analysis lies on the right side, where the YOLOv5 model initially processes the input. Simultaneously, the left side of the flow shows the examination for any plausible character regions utilizing the U-Net model. Each identified region is subsequently reevaluated by the YOLOv5 model to decipher its content. The outcomes from YOLOv5 and U-Net analyses are then merged into a comprehensive list of characters and their respective positions.

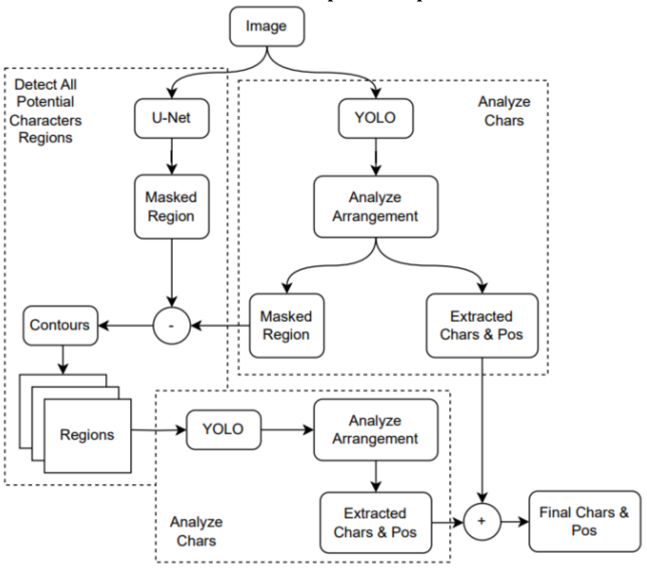


Figure 6. Overall Nusantara script reading based on YOLOv5 and U-Net.

In the 'Analyze Chars' section, a sequence focuses on refining the output of the YOLOv5 model by rearranging character positions. This step is vital because the initial list of character positions provided by YOLOv5 often needs to be sorted. Our sorting approach is straightforward: characters are initially sorted based on their y-axis values, reflecting their height, to align with Nusantara scripts' traditional top-to-bottom reading flow. Additionally, we aim to arrange the script from left to right.

However, sorting characters from left to right becomes challenging when their centroids are not aligned along the same y-axis. To tackle this, our study employs the KMeans clustering algorithm, using y-axis data to deduce the potential number of sentences or lines within the image [32]. We begin by considering the smallest sentence that can be constructed using only two characters. From there, we experiment with clustering the collection of character positions, starting from 2 clusters and moving up to the total number of characters divided by 2. The labels produced by the clustering algorithm are then evaluated using the Silhouette Score, with the highest score indicating the most optimal clustering. Please refer to Algorithm 1 for a more detailed algorithm breakdown.

Once the collection is grouped into clusters and each cluster is marked accordingly, the next step involves normalizing the y-axis values of the characters. Subsequently, all characters sharing the same cluster labels are subjected to a sorting process. Precisely, the algorithm, called **sortByCluster**(locations, clusters), orchestrates the arrangement of character locations within each line, ensuring a seamless left-to-right ordering. The remainder of the algorithm is dedicated to optimizing the visualization of the detected characters. This entails the computation of the necessary rows and columns to present the characters in a user-friendly format. Additionally, all character positions identified by the YOLOv5 model are denoted as YOLO maps.

Algorithm 1 Analyze Characters

```
function ANALYZECHARS($image$)
    $R \leftarrow$ YOLOModel($image$)            ▷ Character locations
    $N \leftarrow$ getCharCount($R$)
    if $N = 0$ then
        $\rightarrow$ None
    $C \leftarrow$ sortYX($R$)            ▷ Sort by y-axis, then x-axis
    if $N < \varepsilon$ then            ▷ We set $\varepsilon$ as 6
        $L \leftarrow 1$            ▷ Consider as 1 line
    else
        $L \leftarrow$ clusterLines($C$)            ▷ Find lines/sentences
    $L \leftarrow$ sortByCluster($C, L$)            ▷ Group into lines
    mean_len $\leftarrow$ avgCharPerLine($L$)
    $r, c \leftarrow$ displayParams($L$, mean_len)            ▷ Row and col
    $\rightarrow C, L, r, c$, mean_len
```

Algorithm 2 Detect Other Characters

```
function DETECTOTHERCHARS($image$)
    $I' \leftarrow$ resize($image$)            ▷ Resize and predict
    $Y \leftarrow$ predictOtherChars($I', M$)
    $Y \leftarrow$ threshold($Y$)
    $Co \leftarrow$ findContours($Y$)            ▷ Contours
    $P \leftarrow []$            ▷ Potential character regions
    for cnt in $Co$ do
        $ce \leftarrow$ centroid(cnt)
        if isPotentialChar($ce, Y$, ref) then
            $x, y, w, h \leftarrow$ bbox(cnt)
            $P$.append($[x, y, w, h]$)
    for $x, y, w, h$ in $P$ do            ▷ Reread in this region
        $T \leftarrow$ crop($I, x, y, w, h$)
        $C, L, r, c$, mean_len $\leftarrow$ AnalyzeChars($T$)
        if $R$ is None then
            continue
        showBoxes($T, C, L, r, c$, mean_len)
```

The subsequent algorithm deals with undetected characters and involves a secondary assessment (Algorithm 2). We employ a U-Net model, labeled as **predictOtherChars**, to anticipate potential character regions. After generating a map of these suggested regions, we extract contours from the map using threshold and findContours functions, denoted as P for potential character regions. These contours provide centroid positions as reference points. Each centroid is cross-referenced with the existing YOLO map. Regions not found in the YOLO map undergo a secondary YOLOv5 model evaluation to ensure no characters were missed in the initial scan. Successfully detected new characters are added to the primary collection of identified characters and their respective positions.

2.3. Data Collection

In this research, an extensive dataset was compiled, encompassing 126 images representing 10 distinct Nusantara Scripts types: Bali, Batak, Bugis, Javanese, Kawi, Kerinci, Lampung, Pallava, Rejang, and Sundanese. This dataset featured various document types and photos, including some from public streets. Part of the dataset is collected from Google Images, while the remainder is obtained through manual handwriting on paper. Manual annotation was carried out on the collected images to ensure appropriate labeling. Subsequently, the dataset was categorized into three subsets for training, validation, and testing. Approximately 64% of the images were allocated for training, 17% for validation, and 19% for testing.

The annotation process employed bounding boxes or rectangles to define object locations, following the methodology outlined in [33]. Figure 7 visually represents this process. Given our training dataset's focus on individual basic letters, vowels, and Panyigeg, we specifically annotated components of the letter 'ing' – the letter 'ha,' the vowel 'I,' and the Panyigeg 'ng.' This approach targets the detection of individual basic letters rather than variations of the 'ing' glyph. Once basic letter classification is completed, an additional step is necessary to connect these distinct basic letters and derive the final reading results.

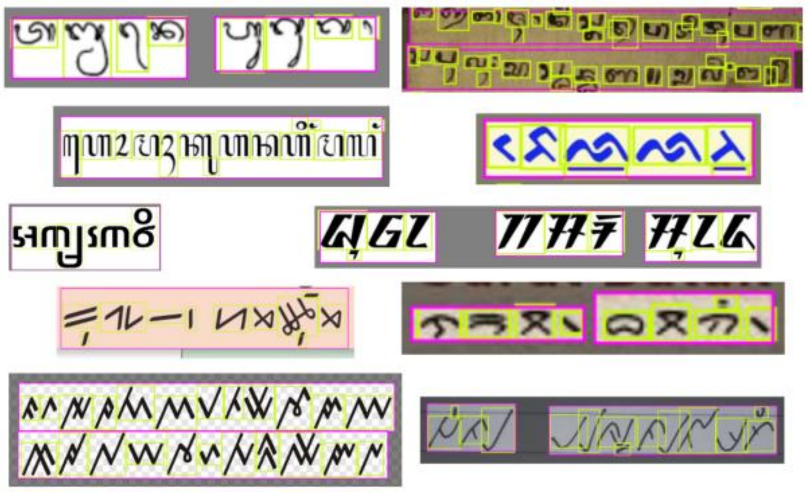


Figure 7. All scripts annotation by considering the position of base characters.

Additionally, we applied data augmentation techniques to diversify the dataset. Each character image underwent three different augmentation methods, including contrast adjustment, grayscale conversion, hue, saturation, brightness, exposure, and noise variations to enrich dataset variation. These augmentations bolstered dataset diversity and robustness for subsequent deep-learning model training.

2.4. Evaluation

There are 2 main types of evaluation that we will use to measure the performance of the models we create: loss performance for character detection and accuracy performance for broader character segmentation prediction. Previous studies that solely utilized a single deep learning approach might have been sufficient for evaluating a specific aspect of performance. However, in this study, two distinct deep learning techniques are employed for distinct purposes: YOLOv5 for individual object detection and U-Net for predicting segments where objects may be distributed. In the first evaluation, box, object, and class loss will be used to measure the character location's bounding box detection performance. The following is a detailed explanation of each loss.

Box Loss evaluates the accuracy of predicting bounding box coordinates and dimensions compared to ground truth boxes, often computed using regression loss functions like MSE or smooth L1 loss, aiding precise object localization. Object Loss, also known as objectness loss, measures how well the model predicts object presence within bounding boxes, often utilizing binary cross-entropy loss to discern object presence accurately. Class Loss assesses the model's accuracy in predicting object categories, quantifying alignment between predicted class probabilities and ground truth labels, usually calculated with classification loss functions like softmax cross-entropy or log loss, enhancing correct object classification.

For the second evaluation, standard sparse categorical cross entropy will be used to measure the predicted region of character locations, where $\text{loss} = -\Sigma\, y_i \log y'_i$. The U-Net character regions prediction model returns small binary maps indicating where the characters may exist. It helps the primary YOLOv5 model to redetect miss-detected characters from the initial reading. This model even returns characters' regions that did not belong to the Nusantara script.

3. RESULT AND DISCUSSION

3.1. YOLOv5 Model Performance

Figure 8 illustrates the comprehensive performance analysis of the YOLOv5 model throughout its training process. The model was trained using 64% of the dataset with a configuration of 16 batches, meaning that in each epoch, 1/16 of the training dataset was processed. The training process spanned 200 epochs, resulting in the entire training dataset being read approximately 12.5 times. As the training progressed, the model exhibited a consistent reduction in its loss value and appeared to approach a state of stability. Specifically, in terms of box loss, the model exhibited a gradual understanding of character locations, resulting in a smooth decline of the loss value to approximately 0.05. When observing object loss, the model displayed some fluctuations, reflecting its efforts to ascertain the presence of objects within the bounding boxes.

3.2. U-Net Model Performance

Figure 9 presents an overview of the U-Net model's performance throughout its training phase. The model demonstrated a steady decline in loss value, indicating effective training. However, a notable observation was the relatively wide gap between the training loss and validation loss, which amounted to approximately 0.2. This discrepancy pointed towards overfitting, a scenario where the model might be excessively fine-tuned to the training data.

Examining the green and red lines in the graph, it becomes evident that both accuracy and validation accuracy exhibited an upward trend over time. This result signifies that the model was progressively improving its prediction accuracy. Notably, the achieved accuracy levels were reasonably high, reaching approximately 0.9 in the region where characters were present and 0.75 in validation accuracy. Remarkably, the model displayed rapid convergence within just 30 epochs.

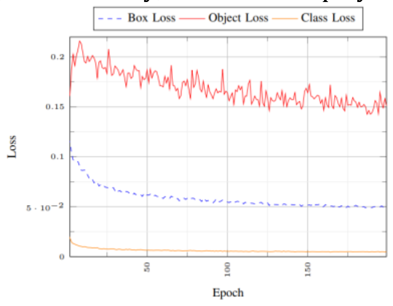


Figure 8. YOLOv5 model performance on Box, Object, and Class Loss during training.

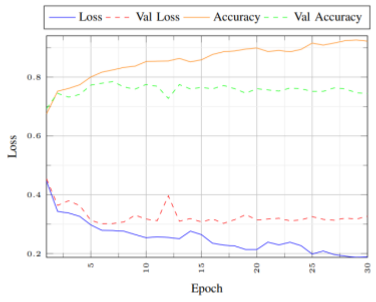
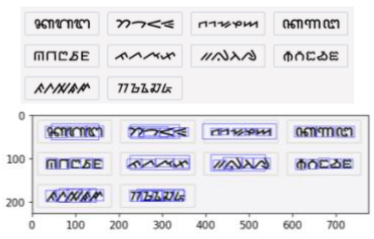


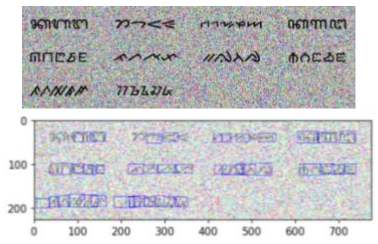
Figure 9. U-Net model performance on character region prediction.

3.3. Experiment Result

This section evaluated our models' performance in reading images with complete Nusantara scripts, inscription, and mixed script content. The capability of reading different Nusantara scripts is demonstrated in Figure 10. Figure 10a displays the result when ten digitalized Nusantara scripts, which are Balinese, Batak, Kerinci (Incung), Javanese, Kawi, Lampung, Bugis (Lontara), Pallawa, Rejang, and Sundanese, are read. These scripts are sourced from aksaradinusantara.com. In Figure 10b, the digitalized scripts are subjected to noise, resulting in the misdetection of 1 Balinese script, 2 Batak scripts, and 1 Kerinci script. Figure 10c features a collection of real-world usage photos depicting 10 Nusantara scripts. Figure 10d showcases the detected scripts, with some characters remaining undetected. Figure 10e presents the final ordered characters, following the processing of all character positions.

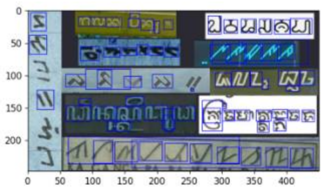


(a) Digital Nusantara scripts detection before (top) and after (bottom).

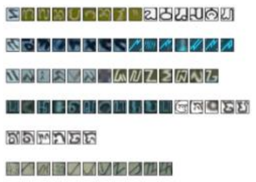


(b) Digital Nusantara scripts detection with noise before (top) and after (bottom).

(c) All Nusantara scripts photos captured from real world.



(d) The detected characters produced by YOLOv5 and U-Net.



(e) Ordered detected characters.

Figure 10. The mixture of Nusantara scripts detection result.

Figure 11 illustrates our model's character detection in a historical photograph of an inscribed stone near Tjiamis before 1900 [34]. The image was initially converted to grayscale for stability (Figure 11a). YOLOv5 detected most characters, with some undetected at the sentence's start (Figure 11b). The characters' positions were then arranged (Figure 11c), with Algorithm 1 determining line numbers and Algorithm 2 optimizing character placement. Figure 11d shows extra detections aided by U-Net mask regions, reintroduced into YOLOv5, highlighting additional characters in yellow boxes. Notably, U-Net complemented YOLOv5, enriching the recognized characters list.

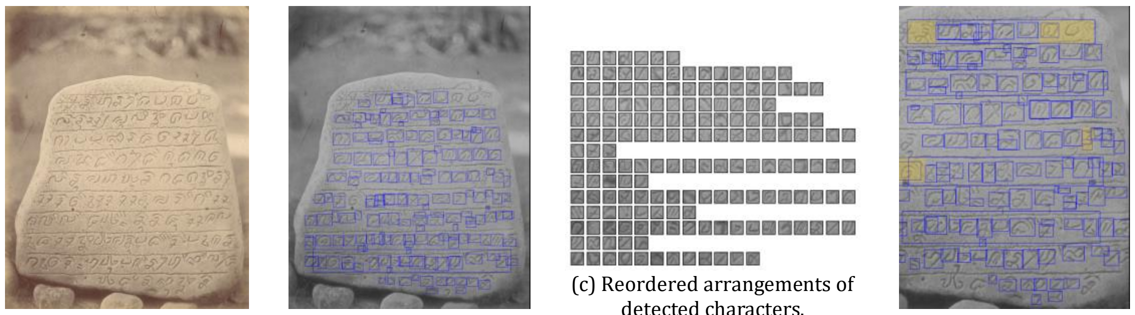

(a) Inscribed stone near Tjiamis. (b) The detected characters produced by YOLOv5. (c) Reordered arrangements of detected characters. (d) Additional characters reproposed by U-Net.

Figure 11. Inscribed stone character detection result.

Figure 12 demonstrates our model's ability to detect a mix of handwritten scripts, including Latin, Javanese, Arabic, Hangeul, Simplified Chinese, Traditional Chinese, and Hiragana characters. Initially, YOLOv5 successfully identified all Nusantara characters and most others, excluding Simplified Chinese characters (Figure 12b). Figure 12c displays the organized character positions. We then applied the U-Net model to detect potential characters, as shown in Figure 12d, with red areas indicating potential character regions. Notably, using Algorithm 2 (Detect-Other-Characters), the U-Net model identified regions missed by YOLOv5, successfully detecting Simplified Chinese characters in this case (Figure 12e).

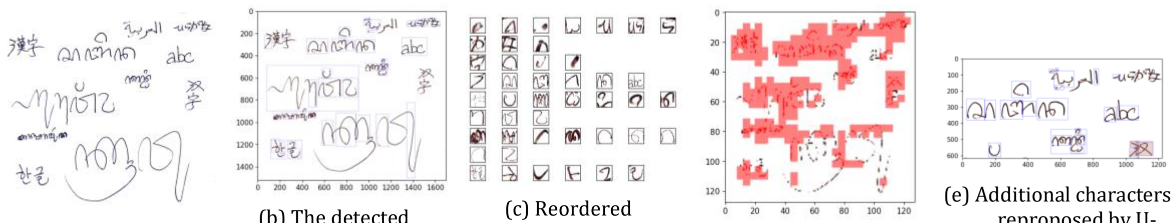

(a) The mixture of handwritten scripts. (b) The detected characters produced by YOLOv5. (c) Reordered arrangements of detected characters. (d) Reproposed detection area by U-Net. (e) Additional characters reproposed by U-Net in red mask.

Figure 12. The mixture of handwritten scripts detection result.

While the capability to detect additional characters may not significantly enhance the accuracy of Nusantara character detection, the outcomes reveal intriguing insights. Although trained exclusively on Nusantara scripts, U-Net models exhibit the capacity to recognize other scripts, even those not included in their specific character sets. It is akin to the U-Net model being trained solely for Nusantara script detection, yet the results indicate its ability to identify Latin, Arabic, Korean, Japanese, and Chinese scripts.

3.4. Discussion

YOLOv5 is a character detection model that has strong capabilities but struggles with proper Nusantara script arrangement. It has a box loss of approximately 0.05, object loss approaching 0.15, and class loss approaching 0.005, suggesting it may not always detect all characters. The U-Net model consistently increases accuracy but shows stagnant improvement in validation accuracy on generating masks, suggesting overfitting. In experiments, YOLOv5 may miss some Nusantara script characters, around 3% of documents. This issue may arise due to the presence of low-resolution images and complex backgrounds, which are frequently encountered in old manuscripts. In the initial test with digital Nusantara scripts, the model exhibited flawless script detection. However, when noise was introduced, the model's detection capability decreased by 10%. Even the U-Net model did not yield significant improvements on the noisy dataset, indicating that noise indeed impacts the U-Net model's performance. The U-Net model addressed some misses, improving only 1.2% of overall documents, occasionally identifying characters from other scripts. This highlights the efficiency of character location detection using YOLOv5 and U-Net for Nusantara scripts.

4. CONCLUSION

In this study, we developed Nusantara script detection models using YOLOv5 and U-Net. However, YOLOv5's character arrangement does not conform to proper Nusantara script writing conventions. We introduced a sentence prediction function using clustering techniques and an arrangement algorithm for systematic character organization to address this. While YOLOv5 performed well, some characters went undetected, prompting us to incorporate the U-Net model, which improved detection rates, including characters from untrained scripts.

Experimental results show YOLOv5 achieved a character location detection loss of approximately 0.05 for box loss, 0.15 for object loss, and 0.005 for class loss for , while the U-Net model demonstrated an accuracy ranging from 75% to 90% in predicting character regions. Although YOLOv5 does not achieve perfect Nusantara script detection, integrating the U-Net model increased detection by 1.2%. While this research advanced Nusantara script detection, further work is needed to finalize character reading. Future studies can explore refining detection models, increasing number of dataset, including character direction and sentence degree detection, to enhance reading results.

REFERENCES

[1] P. K. Charles, V. Harish, M. Swathi, and C. H. Deepthi, "A review on the various techniques used for optical character recognition," International Journal of Engineering Research and Applications, vol. 2, no. 1, pp. 659-662, 2012.

[2] G. Nagy, S. Seth, and M. Viswanathan, "A Prototype Document Image Analysis System for Technical Journals," Computer, vol. 25, no. 7, pp. 10–22, 1992.

[3] D. R. Dickson and K. Nusair, "An HR perspective: The global hunt for talent in the digital age," Worldwide Hospitality and Tourism Themes, vol. 2, no. 1, pp. 86–93, 2010. doi: 10.1108/17554211011012612.

[4] J. Lo Bianco, "The importance of language policies and multilingualism for cultural diversity," International Social Science Journal, vol. 61, no. 199, pp. 37–67, 2010. doi: 10.1111/j.1468-2451.2010.01747.x.

[5] Y. Yamashita, K. Higuchi, Y. Yamada, and Y. Haga, "Classification of handprinted Kanji characters by the structured segment matching method," Pattern Recognition Letters, vol. 1, no. 5-6, pp. 475-479, 1983.

[6] G. Lee, J. H. Lee, and J. Yoo, "Multi-level post-processing for Korean character recognition using morphological analysis and linguistic evaluation," Pattern Recognition, vol. 30, no. 8, pp. 1347-1360, 1997.

[7] C. L. Liu, F. Yin, D. H. Wang, and Q. F. Wang, "Online and offline handwritten Chinese character recognition: benchmarking on new databases," Pattern Recognition, vol. 46, no. 1, pp. 155-162, 2013.

[8] M. Avadesh and N. Goyal, "Optical character recognition for Sanskrit using convolution neural networks," in 2018 13th IAPR International Workshop on Document Analysis Systems (DAS), pp. 447-452, 2018. doi: 10.1109/DAS.2018.27.

[9] A. Prasetiadi, J. Saputra, I. Ramadhanti, A. D. Sripamuji, and R. R. Amalia, "Minimalist DCT-based Depthwise Separable Convolutional Neural Network Approach for Tangut Script," Journal of Dinda: Data Science, Information Technology, and Data Analytics, vol. 3, no. 2, pp. 59-64, 2023.

[10] S. Alghyaline, "A Printed Arabic Optical Character Recognition System using Deep Learning," Journal of Computer Science, vol. 18, no. 11, pp. 1038–1050, 2022. doi: 10.3844/jcssp.2022.1038.1050.

[11] B. Kataria and H. B. Jethva, "CNN-Bidirectional LSTM Based Optical Character Recognition of Sanskrit Manuscripts: A Comprehensive Systematic Literature Review," International Journal of Scientific Research in Computer Science, Engineering and Information Technology, pp. 1362–1383, 2019. doi: 10.32628/cseit2064126.

[12] A. W. Mahastama and L. D. Krisnawati, "Optical character recognition for printed Javanese script using projection profile segmentation and nearest centroid classifier," in 2020 Asia Conference on Computers and Communications (ACCC), pp. 52–56, 2020. doi: 10.1109/ACCC51160.2020.9347895.

[13] M. H. Faishal, M. D. Sulistiyo, and A. F. Ihsan, "Javanese Script Letter Detection Using Faster R-CNN," Indonesian Journal of Artificial Intelligence and Data Mining, vol. 6 no. 2, 243-251, 2023

[14] N. Suciati, N. P. Sutramiani, and D. Siahaan, "LONTAR_DETC: Dense and High Variance Balinese Character Detection Method in Lontar Manuscripts," IEEE Access, vol. 10, pp. 14600-14609, 2022.

[15] B. Gašparović, G. Mauša, J. Rukavina, and J. Lerga, "Evaluating YOLOv5, YOLOv6, YOLOv7, and YOLOv8 in Underwater Environment: Is There Real Improvement?," in 2023 8th International Conference on Smart and Sustainable Technologies (SpliTech), pp. 1-4, June 2023.

[16] T. A. N. Dang and D. T. Nguyen, "End-to-end information extraction by character-level embedding and multi-stage attentional U-Net," arXiv preprint arXiv:2106.00952, 2021.

[17] A. Prasetiadi, J. Saputra, I. Kresna, and I. Ramadhanti, "Deep Learning Approaches for Nusantara Scripts Optical Character Recognition," IJCCS (Indonesian Journal of Computing and Cybernetics Systems), vol. 17, no. 3, 2023.

[18] D. Ghosh, T. Dube, and A. Shivaprasad, "Script Recognition-a review," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 12, pp. 2142–2161, 2010. doi: 10.1109/TPAMI.2010.30.

[19] E. Alfian, "Penggunaan Unsur Aksara Nusantara Pada Huruf Modern," Jurnal Komunikasi Visual, vol. 7, no. 1, pp. 42–48, 2014.

[20] P. T. Daniels, "Fundamentals of Grammatology," Journal of the American Oriental Society, vol. 119, no. 4, pp. 727–731, Oct.–Dec. 1990. doi: 10.2307/602899.

[21] J. Chen, M. Xie, Z. Xing, C. Chen, X. Xu, L. Zhu and G. Li, "Object detection for graphical user interface: Old fashioned or deep learning or a combination?," in Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Nov. 2020, pp. 1202-1214.

[22] V. Lempitsky, P. Kohli, C. Rother, and T. Sharp, "Image Segmentation with a Bounding Box Prior," in International Conference on Computer Vision (ICCV), IEEE, pp. 277–284, 2009.

[23] J. Son, M. Baek, M. Cho, and B. Han, "Multi-object tracking with quadruplet convolutional neural networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5620-5629, 2017.

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I, pp. 21-37, 2016.

[25] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A Review of Yolo algorithm developments," Procedia Computer Science, vol. 199, pp. 1066-1073, 2022.

[26] P. Bharati and A. Pramanik, "Deep learning techniques—R-CNN to mask R-CNN: a survey," in Computational Intelligence in Pattern Recognition: Proceedings of CIPR 2019, pp. 657-668, 2020.

[27] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, K. Michael, X. Tao, J. Fang, I. Imyhxy, L. Lorna, Y. Zeng, C. Wong, A. V, D. Montes, Z. Wang, C. Fati, J. Nadar, Laughing, D. UnglvKitDe, V. Sonck, T. Tkianai, Y. YxNONG, P. Skalski, A. Hogan, D. Nair, M. Strobel, and M. Jain, "ultralytics/yolov5: v7.0-yolov5 sota realtime instance segmentation," Zenodo, 2022.

[28] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788, 2016.

[29] C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao, "Scaled-yolov4: Scaling cross stage partial network," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 13029-13038.

[30] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in 18th International Conference on Pattern Recognition (ICPR'06), 2006, pp. 850-855.

[31] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," arXiv:1505.04597, 2015.

[32] A. M. Ikotun, A. E. Ezugwu, L. Abualigah, B. Abuhaija, and J. Heming, "K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data," Information Sciences, 2022.

[33] R. H. Pramestya, "Deteksi dan Klasifikasi Kerusakan Jalan Aspal menggunakan Metode YOLO berbasis Citra Digital," 2018. [Online]. Available: http://repository.its.ac.id/id/eprint/59044.

[34] I. van Kinsbergen, "Inscribed stone at Kawali near Tjiamis," in KITLV Digital Image Collection, KITLV, Before 1900. [Online]. Available: https://digitalcollections.universiteitleiden.nl/view/item/770870.