Development of mobile-based Batak script recognition application using YOLOv8 algorithm

Iustisia Natalia Simbolon, Herimanto, Ranty Deviana Siahaan, Samuel Adika Lumbantobing, Grace Natalia Br Sitepu
Departments of Informatics, Faculty of Informatics and Electrical Engineering, Institut Teknologi Del, Toba, Indonesia

Article Info

Article history:

Received Aug 6, 2024
Revised Mar 18, 2025
Accepted Jul 1, 2025

Keywords:

Confidence score
Image preprocessing
Object detection
Training
YOLOv8

ABSTRACT

The Batak people are one of the ethnic groups that pass down many values and traditions to each generation, including the written tradition known as the Batak script. The Batak Toba people, in particular, have the Batak Toba script as part of their local wisdom that needs to be preserved and maintained. However, the use of the Batak script has significantly declined in the current era. To prevent the loss of this heritage, preservation through technology is necessary. This research utilizes a deep learning approach using the YOLOv8 algorithm to detect images of script objects, provide the coordinates of the script locations, and perform object recognition based on the dataset. The final result of this research is an Android-based application that can detect the Batak Toba script in real time and upload images. The research process involves experiments on several hyperparameters, such as epochs with a value of 200, confidence threshold, and IoU with a value of 0.5. The model evaluation shows excellent results, with a precision of 0.945, recall of 0.902, mAP@0.5 of 0.954, and a high confidence score from the application's detection.

Corresponding Author:

Iustisia Natalia Simbolon
Departments of Informatics, Faculty of Informatics and Electrical Engineering, Institut Teknologi Del
Sisingamangaraja Street, Sitoluama, Laguboti 22381, Toba, Indonesia
Email: iustisia.simbolon@del.ac.id

1. INTRODUCTION

The Batak Tribe, particularly the Batak Toba, is renowned for its written tradition heritage, specifically the Batak Toba script. The Batak script is classified as an abugida, a type of phonetic writing where each language can be accurately represented. The writing of Batak letters is broadly divided into two categories: "*Ina ni Surat*" and "*Anak ni Surat*". Unfortunately, the use of this script has significantly declined [1]. Therefore, it is crucial to preserve it using technology, particularly in the rapidly evolving field of machine learning.

This research holds significant importance for several interrelated reasons. First, it aims to preserve cultural heritage by developing a mobile application using machine learning algorithms like YOLOv8 to facilitate the learning and use of the Batak Toba script among younger generations and the wider community, thus contributing to the continuity of the Batak people's cultural and historical identity. Second, the application has high educational value by providing an interactive and engaging method to learn the Batak Toba script, which may not be available in traditional educational resources, thereby enhancing cultural literacy and understanding of the Batak script and helping users understand the meaning and significance of a Batak script. Third, the use of YOLOv8 in this context demonstrates technological advances in preserving the Batak script, as the application helps users understand the meaning of a Batak script, whose comprehension has begun to fade in the current era [2], [3]. Fourth, the practical implementation of machine learning in real-time object detection not only provides a relevant case study in AI applications but also showcases the potential of technology for cultural applications [4]. Lastly, by making the Batak Toba script more accessible through a mobile application, this research contributes to the increased accessibility and use, supporting long-term goals in the preservation and revitalization of Batak culture [5], [6].

By addressing these aspects, this research not only aims to preserve an important part of Batak heritage but also contributes to the broader discourse on the role of technology in cultural preservation. The integration of YOLOv8 in this context highlights the innovative use of machine learning in tackling cultural and educational challenges. The objectives of this research include i) implementing the YOLOv8 algorithm in a real-time Batak script recognition application and image import to facilitate the recognition of root words and affixes; ii) measuring the accuracy of the YOLOv8 algorithm in detecting objects through the evaluation of the Batak Toba script model; and iii) integrating the Batak Toba script object detection results with a mobile application using TensorFlow Lite.

The YOLO algorithm is able to detect objects in real-time well. The development of YOLOv4 increases the AP and FPS values of YOLOv3 by 10% and 12% [3]. This is different from YOLO and YOLOv2 which are not effective in detecting small targets so that multi-scale detection is added to YOLOv3 [7]. The YOLOv4 algorithm introduced by Alexey Bochkovskiy has a speed of up to 45 fps. However, the YOLOv7 algorithm is one of the object detection algorithms that can be done in real-time with high efficiency and accuracy. The YOLOv7 algorithm has an accuracy of 56.8% and has a high detection speed, reaching 5-160 FPS [8] even in situations where there is more than one object in the image. Liu *et al.* [9] YOLOv8, FPS consistently above 300.

Therefore, this study will conduct analysis and implementation of real-time object detection using the YOLOv8 algorithm. In this study, the object to be detected is Batak script. This study will implement the YOLOv8 algorithm in object detection and is expected to be carried out in real-time with high efficiency and accuracy, and allows the system to quickly identify script objects. This study will also analyze YOLOv8 hyperparameters to ensure that the resulting model provides optimal detection. Hyperparameter settings will be adjusted to the dataset used in the experiment to maximize model performance in object detection. Hyperparameter tuning is carried out to find the most optimal model accuracy value in real-time object detection. The model with optimal hyperparameters will be implemented into a mobile-based application and is expected to automate object detection.

2. RELATED RESEARCH

In previous research, the detection of the Batak Toba script was conducted using the CNN algorithms to detect each given image input and assign a class name to each detected object [1]. However, this research only reached the stage of script detection without implementation in an application, and it only detected "ina ni surat" (root words), without detecting "anak ni surat" (affixes). Pratama *et al.* [10], the recognition and translation of the Batak Toba script were successfully carried out using single shot detection by implementing the CNN architecture. However, there is the detection of repeated objects in the image using single shot detection which affects the detection results.

This research uses the YOLOv8 algorithm, which is more efficient and accurate in detecting the Batak Toba script, including affixes. YOLOv8 implements a convolutional neural network (CNN)-based detection method [11]. As the most common representation of single-stage object detection algorithms, YOLOv8 is a neural network-based algorithm used to identify and determine object locations. YOLOv8 uses a single CNN model to detect end-to-end objects. Single-stage detectors treat object detection as a regression/classification problem using a unified framework to obtain labels and locations directly. These detectors link image pixels directly to bounding box coordinates and class probabilities [12]. This is done by proposing prediction boxes directly from the input image without a region proposal step [13]. The YOLOv8 algorithm takes the entire image as input into the network structure and directly regresses the bounding box locations while classifying objects into appropriate categories in the output layer. YOLOv8 was chosen because it can detect smaller objects with higher resolution (608 x 608 pixels) compared to YOLOv3, and it processes images at a speed of 155 frames per second [14]. The YOLOv8 approach allows for the detection of root words and affixes and results in an Android application for real-time detection. The YOLOv8 algorithm uses nine anchor boxes to detect a wider variety of object shapes and sizes [15].

3. METHOD

This phase involves explaining the research chronologically, including the research design, research procedures, testing methods, and data acquisition.

3.1. YOLOv8 model

Object detection is one of the crucial tasks in the field of computer vision, with broad applications ranging from security surveillance to autonomous vehicles. A popular approach for object detection is the you only look once (YOLO) model [16], known for its ability to detect objects in real time with high accuracy. YOLOv3, introduced by Redmon and Farhadi, is one of the most widely used versions of YOLO. This model uses residual blocks, multi-scale predictions, and a deeper Darknet-53 backbone network [17]. These enhancements allow YOLOv3 to achieve a better balance between speed and accuracy compared to YOLOv2 [2]. A study by Chen *et al.* [12] demonstrated that YOLOv3 is effective in detecting objects in medical images, such as tumors in ultrasound images [18].

YOLOv5, although not released by the original authors of YOLO, has attracted significant attention in the research community. Research by Zhang *et al.* [19] highlights several advantages of YOLOv5, such as smaller model size, faster inference speed, and ease of use with the PyTorch framework [19]. YOLOv5 also introduces several model size variants (s, m, l, x) that can be tailored to specific needs. Another study by Shen et al. [20] showed that YOLOv5 can be used for object detection in complex traffic environments with high accuracy [20]. According to research by Wang *et al.* [8] YOLOv7 is the fastest and most accurate YOLO model to date. This model adopts technological innovations such as model scaling, re-parameterized convolution, and the use of an efficient layer aggregation network (ELAN). The study results indicate that YOLOv7 achieves superior performance across various object detection benchmarks, with an optimal balance between speed and accuracy [8]. Research by Chen *et al.* [12] demonstrated that YOLOv7 could be used for object detection in industrial environments, such as defect detection on production lines [8].

YOLOv8 is the latest version of the YOLO object detection model. This version introduces a new neural network architecture that leverages the feature pyramid network (FPN) and path aggregation network (PAN) [21]. Additionally, YOLOv8 comes with new labeling tools that simplify the annotation process, including features like auto-labeling, labeling shortcuts, and customizable hotkeys. These improvements make the image annotation process for model training easier. However, further research is needed to evaluate YOLOv8's performance in real-world applications.

The development of various YOLO versions demonstrates significant progress in object detection, with continuous improvements in speed and accuracy. YOLOv8, as the latest version, offers various enhancements that make it a more efficient tool for object detection. Further research is necessary to evaluate YOLOv8's performance in real-world applications and compare it with previous versions.

3.2. YOLOv8 architecture

YOLOv8 comprises three main modules: the backbone, neck, and head, as illustrated in Figure 1. YOLOv8 maintains the basic structure of YOLOv5 but replaces the C3 module with the C2f module, which integrates the CSP and ELAN concepts from YOLOv7 [21]. The C2f module enhances gradient flow, improving the capture and representation of image features. YOLOv8 continues to use the SPPF module at the end of the backbone, consisting of three layers of 5×5 Maxpool, ensuring accurate object detection at various scales. The integration of C2f and SPPF keeps YOLOv8 lightweight and suitable for real-time applications [21].

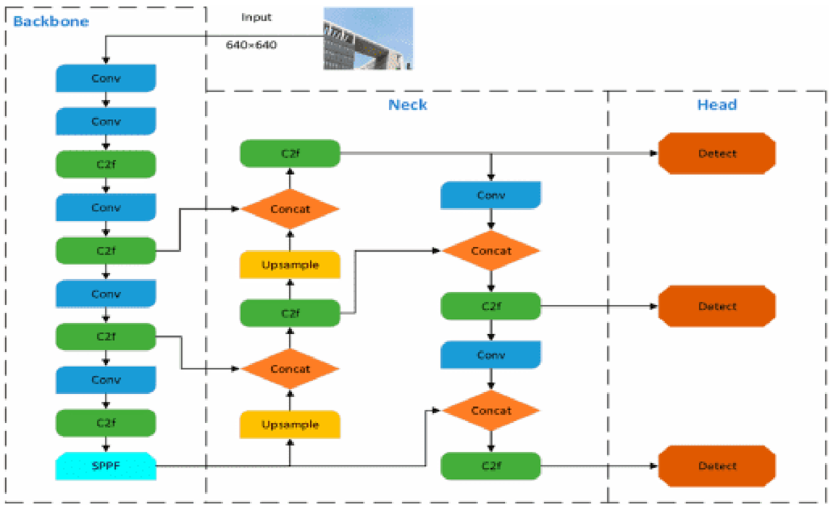


Figure 1. YOLOv8 architecture [20]

In the neck section, YOLOv8 uses PAN-FPN to combine and utilize information from different feature scales [21]. The C2f module ensures that important information is not missing during merging. YOLOv8 also adopts a decoupled head structure, as seen in YOLOx, to improve accuracy by separating classification and regression tasks. This approach ensures that YOLOv8 remains efficient and accurate for real-time object detection.

By separating the classification and localization branches, YOLOv8 reduces conflicts between these tasks, enhancing overall accuracy and efficiency in handling various types of objects in images [22]. This decoupled head approach ensures consistent and accurate results [21].

3.3. Batak Toba script detection mechanism

The development of a Batak Toba script recognition application involves several stages: preprocessing, labeling, and training data to create a model using the YOLOv8 algorithm. The detection application will be integrated with the YOLOv8 model and will include features for capturing images via a camera and uploading images. When you point the camera at the Batak Toba script, the application will automatically detect the characters and provide their corresponding spelling. The application operates through several stages to identify script objects, detect bounding boxes, and classify script characters. Details of these stages are shown in Figure 2.

In Figure 2, the first confidence score is found in the first channel, and the first bounding box with center coordinates (x, y), width (w), and height (h) are shown in channels two through five, marked with purple grids. Meanwhile, the second confidence score is in the sixth channel, and the second bounding box with center coordinates (x, y), width (w), and height (h) are shown in channels seven through ten, marked with green grids, and the white grid indicates the number of detected object classes.

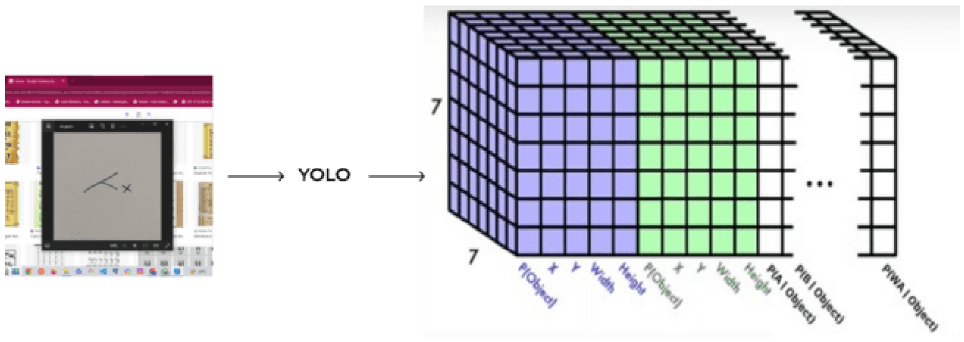


Figure 2. Detection of Batak Toba script

3.3.1. Bounding box prediction mechanism

The model predicts bounding boxes using the following mechanism:

a) The input image is divided into a grid of size S×S. Each cell in the grid is responsible for predicting whether an object's center is within that cell.
b) Feature extraction is performed using Convolutional Layers, where the input image is processed through a series of convolutional layers to extract its features. Figure 3 illustrates how the output from the neural network is organized into a grid, with each cell storing information needed to detect objects, including bounding box predictions, confidence scores, and class category predictions. Each grid cell contains 2 bounding boxes and 2 confidence values.
c) In Figure 3, the first confidence score is in the first channel, and for the first bounding box, the center coordinates (x, y), width (w), and height (h) are shown in channels two through five, marked with purple grids. The second confidence score is in the sixth channel, and for the second bounding box, the center coordinates (x, y), width (w), and height (h) are shown in channels seven through ten, also marked with purple grids.
d) Bounding box predictions are made as follows: Each cell in the grid predicts several bounding boxes (e.g., N bounding boxes per cell). Each bounding box is described by five parameters: center coordinates (x, y), width (w), height (h), and confidence score. The coordinates (x, y) are normalized relative to the cell size, while (w, h) are normalized relative to the size of the entire image. The confidence score reflects the model's belief in the presence of an object within the bounding box and the accuracy of the bounding box.

e) Confidence score formulation

In object detection, the model outputs bounding box predictions marking the object's location along with a confidence score. The confidence score is calculated as follows:

$$Confidence = Pr\,(Object) \times IoU_{pred}^{truth} \tag{1}$$

Pr(Object) represents the probability that there is an object within the predicted bounding box. This probability reflects the model's confidence that the bounding box indeed contains an object.

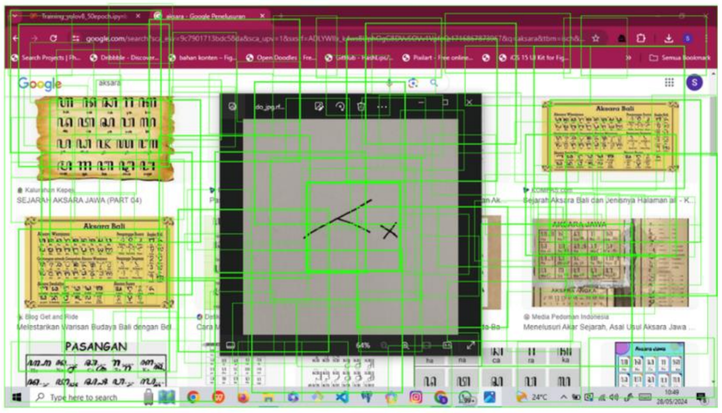


Figure 3. Bounding box prediction

IoU_{pred}^{truth} is a metric used to measure the overlap between two bounding boxes, which is crucial for improving detection accuracy. The presence of multiple bounding boxes indicates initial detection results by the model, and IoU helps evaluate accuracy by comparing predictions with ground truth bounding boxes. IoU is used in the non-maximum suppression (NMS) process to remove overlapping boxes by selecting the box with the highest confidence score, and in setting thresholds to filter out less accurate predictions. Therefore, IoU plays a vital role in achieving more accurate object detection and reducing duplicate bounding boxes.

f) Threshold and NMS

A threshold is applied to the confidence score of each prediction. Predictions with confidence scores below the threshold are ignored, reducing the number of bounding boxes considered. After applying the threshold to filter out predictions with low confidence scores, NMS is used to remove overlapping bounding boxes. Among multiple bounding boxes indicating the same object, NMS retains only the box with the highest confidence score and discards the others. This process results in more accurate object detection and reduces duplication, producing cleaner and clearer results, as illustrated in Figure 4.

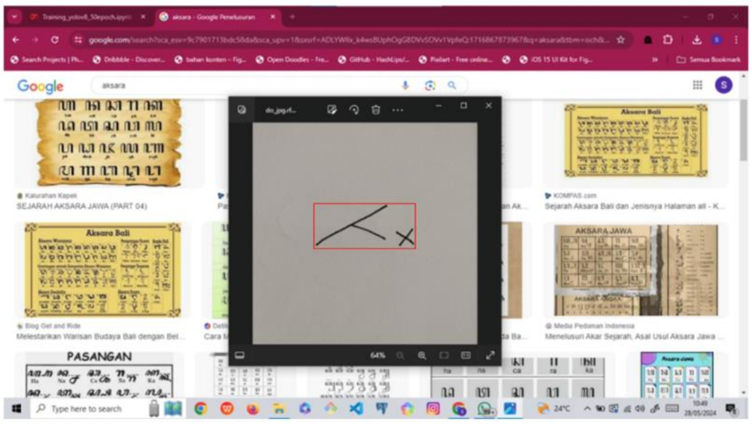


Figure 4. Determination of bounding box size and coordinates

3.3.2. Character identification prediction mechanism

After the bounding box detection process is complete, the next step is character identification. Character identification is the process of determining the class category of the object detected within the bounding box.

a) Conditional class probability: each grid cell also predicts a probability distribution for the potential object classes within the bounding box. For example, if there are N possible classes, each cell predicts an N-sized vector where each element represents the probability of the object belonging to that class. In Figure 2, the white grid indicates the number of detected object classes, which is 181 classes in this study.
b) Class prediction in each cell: in Figure 5, class probabilities are calculated using the softmax function on the class output to obtain a valid probability distribution. These probabilities are conditioned on the grid cell containing the object.
c) Multiplying class probability and confidence score: during inference, the class probability of each bounding box is multiplied by the confidence score to obtain the final confidence score for each class, highlighted by the thickest bounding box at the end of Figure 5.

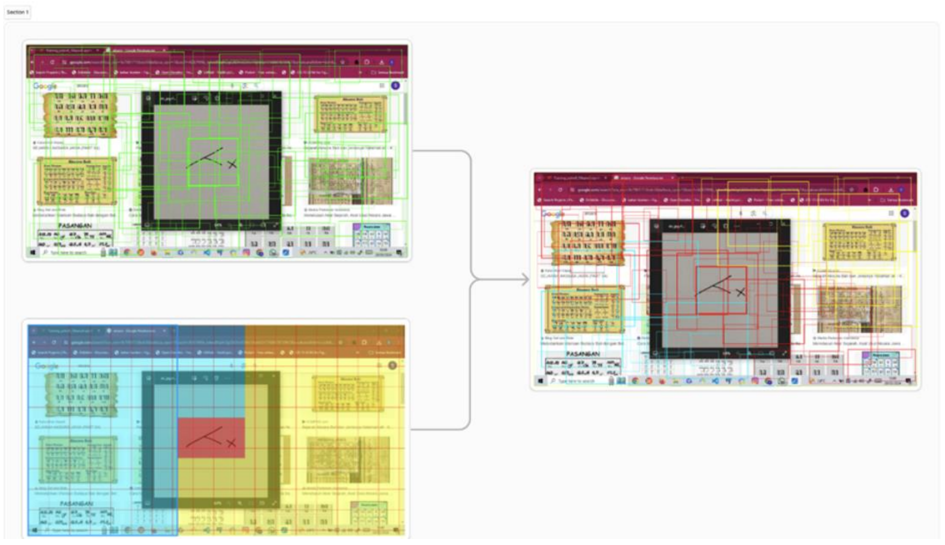


Figure 5. Character object identification process

d) Determination of class and object class score: this class score reflects the probability of the class occurring within the bounding box and how well the bounding box prediction matches the object. The class with the highest score is then selected as the final class of the object within that bounding box.

3.4. Development of Batak script detection model with TensorFlow lite

TensorFlow Lite is one of the most popular frameworks for machine learning [23]. TensorFlow is an interface and an implementation for executing machine learning algorithms presented in 2015 [24]. Tensorflow Lite is a free deep learning framework that enables developers to build and deploy machine learning models [25]. Developing an Android application requires features that enable users to connect with the camera and capture images for the system to test. This application will be built using Android Studio and Kotlin, with TensorFlow Lite as the framework to implement the machine learning model for image recognition.

3.5. Experiments

The research begins with studying relevant literature and collecting image data of the Batak Toba script to be used. Once the image data is gathered, the next step is image processing aimed at improving the quality and variety of the dataset or preparing the data in a suitable format, including augmentation: image rotation and brightness adjustment, resizing, and data labeling.

After processing the images, the data is divided into three sets: validation (20%), training (70%), and testing (10%). In this study [22], the dataset division used was 70: 20: 10, which is the optimal dataset division ratio according to research that has been conducted, with Accuracy (mAP) results of 0.9951 and training time results of 15 minutes 55 seconds. The training data is used to train the object detection model using the YOLOv8 algorithm. Once training is complete, the model is evaluated using the validation set to assess its initial performance. Model evaluation ensures the model works well before final testing. Testing uses the test set to assess the model's overall performance and select the best model based on evaluation results. The optimal model is subsequently converted to the TFLite format for integration and deployment within the application. The final step is developing an application that uses the TFLite model for object detection, completing the project workflow.

3.5.1. Dataset

In this study, the dataset collected consists of 181 classes with 2,408 annotations. There are 1,857 labels for training data, 532 labels for validation data, and 254 labels for testing data. The collected dataset undergoes preprocessing, labeling, and dataset splitting.

a) Image preprocessing

The collected script images are resized to 640×640 pixels. Figure 6 shows the results of the resizing process. After resizing, the next step is augmentation: rotation with scales of -2°, -4°, -6°, -8°, -10°, 10°, 8°, 6°, 4°, 2° as shown in Figure 7, and brightness adjustment with a scale of 0.7 as shown in Figure 8.

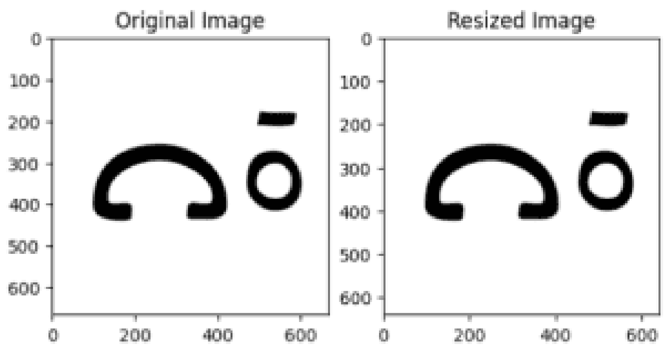


Figure 6. Resizing of script image dimensions

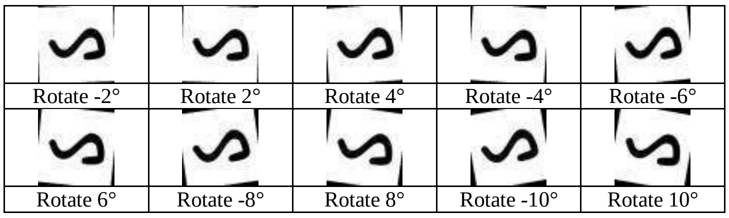


Figure 7. Rotation of Batak Toba script mages

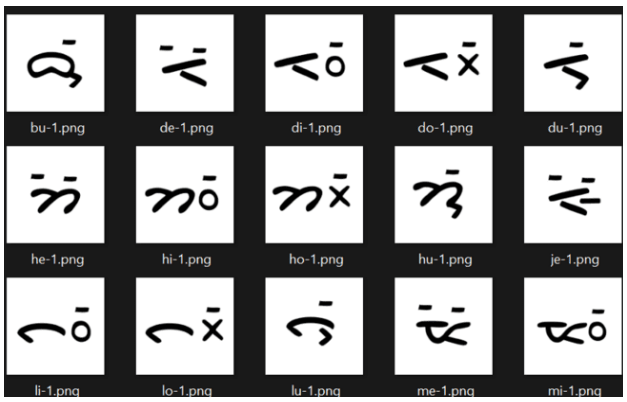


Figure 8. Brightness adjustment of Batak Toba script images

b) Dataset labeling

In this study, labeling was performed using the Roboflow platform, involving the creation of rectangular bounding boxes around each script object. Upon completion, a file containing all the label classes, called classes.txt, was generated. The results from this stage will proceed to the training process.

c) Dataset splitting

The details of the dataset division in this study are shown in Table 1. The dataset is split into 70% for training, 20% for validation, and 10% for testing. This distribution will proceed to the training phase to obtain the optimal model.

Table 1. Dataset splitting

Dataset splitting	Number of images
Train Set 70%	1857
Valid Set 20%	532
Test Set 10%	254
Total	2643

3.5.2. Hyperparameter evaluation

The data validation process in this study involves determining crucial hyperparameters, such as the IoU threshold, at the initial stage. Following this, the validation results are evaluated, and a thorough analysis is conducted during the decision-making phase. If the mean average precision (mAP) achieves a high and stable performance level according to the standard value, a reevaluation with different hyperparameter values is conducted to achieve optimal results. A standard value is considered good at 0.5, and values above 0.5 are deemed very good.

4. RESULTS AND DISCUSSION

4.1. Training data results and discussion

After gathering image data and splitting it, the dataset undergoes the training process. The results are shown in Tables 2 and 3. Based on Table 3, the experiment results with various epochs show that the parameter with 200 epochs has a higher mAP value compared to other epoch parameters, specifically 0.954. A high mAP value indicates a good balance between precision and recall. This means the model is generally better at detecting and recognizing Toba script objects under various conditions. Besides mAP, further improvements in precision and recall demonstrate that the model is more reliable in detecting Toba script.

Figure 9 shows the training results from Table 3 with the x-axis showing the number of epochs and the y-axis of the first image the range of mAP values, the second image the precision and the third the recall. The Toba script model shows significant improvements in precision, recall, and mean Average Precision (mAP) metrics at a threshold of 0.5 during training. Precision and recall values increased from 0.0 to nearly 0.9 at epoch 150, then stabilized until epoch 200, indicating the model reached optimal accuracy in avoiding false positives and detecting almost all objects. The mAP value also increased from 0.0 to nearly 0.95 at epoch 150 and stabilized until epoch 200, reflecting good overall model performance in object detection. These results indicate the model effectively learned over 200 epochs, with improved accuracy and good pattern recognition in the data.

Table 2. Training time process based on epoch size

Image size	Epoch	Training duration	Description
640 x 640	50	0.7 hours	Training successful.
640 x 640	100	1.5 hours	Training successful
640 x 640	150	2.9 hours	Training successful
640 x 640	200	2.9 hours	Training successful

Table 3. Experiment results with various epochs

Epoch	Precision	Recall	mAP@.5
50	0.763	0.803	0.858
100	0.841	0.874	0.925
150	0.856	0.896	0.952
200	0.945	0.902	0.954

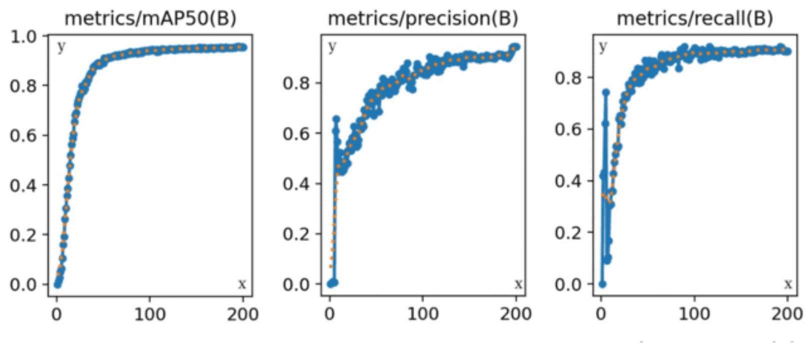
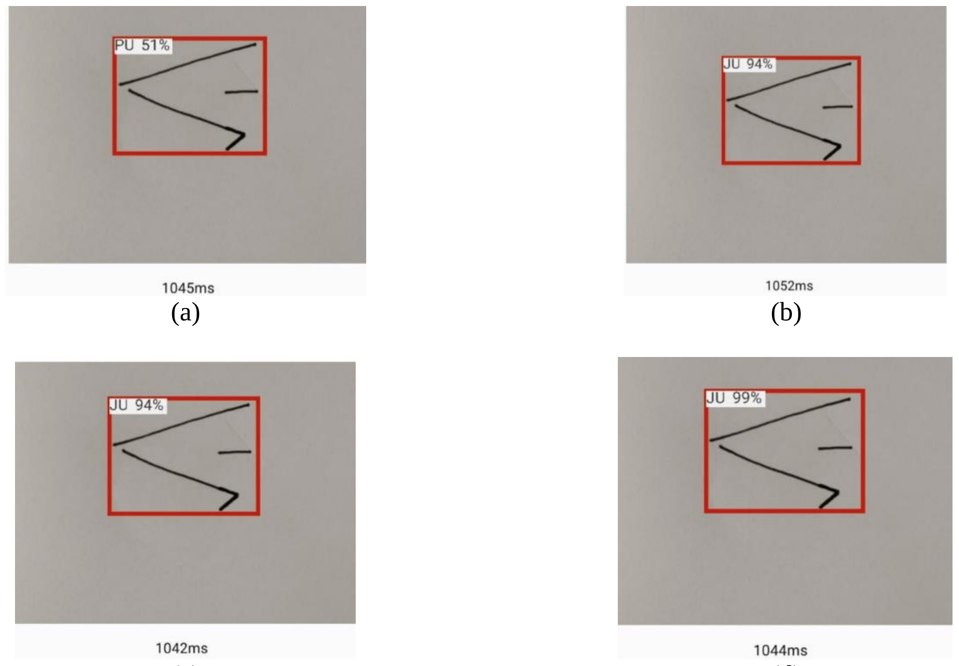


Figure 9. Hyperparameter training dataset evaluation results

4.2. Model evaluation results and discussion

Precision, recall, and mAP values during training are influenced by various factors, including the number of epochs, the quality and quantity of training data, preprocessing and augmentation techniques, and hyperparameters. Models trained with a higher number of epochs tend to have better precision, recall, and mAP because they have more opportunities to learn and optimize their performance from the available data.

The Toba script object detection experiments using models with 50, 100, 150, and 200 epochs are presented in Figure 10. The model used in Figure 10(a) has 50 epochs. The results show incorrect object detection, where the correct label should be "PU" instead of "JU". For images with 100 epochs Figure 10(b) and 150 epochs Figure10(c), the detection results both show a confidence value of 94%, but with different detection speeds, with the model trained for 150 epochs detecting faster than the 100-epoch model. For the image with the 200 epoch Figure10(d) model, the detection results show a very good detection accuracy of 99%.



(a) (b) (c) (d)

Figure 10. Experiment results with various epochs: (a) 50 epoch, (b) 100 epoch, (c) 150 epoch, and (d) 200 epoch

The research experiments using confidence threshold parameters of 0.7 and 0.4 are shown in Tables 4 and 5. The comparison of detection results with these confidence threshold values proves that the higher the threshold, the more accurate and better the model detects Batak Toba script.

Table 4. Experiment with confidence threshold parameter 0.4

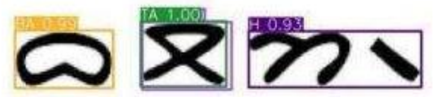
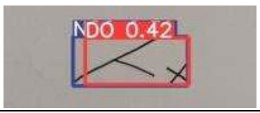
Detection results	Remarks
	Detection results by the 200-epoch model with a confidence threshold parameter of 0.4. There are still two bounding boxes with different classes, TA and T.
	Detection results by the 200-epoch model with a confidence threshold parameter of 0.4. There are still two bounding boxes with different classes, NA and DO.

Table 5. Various cases experiment with YOLOv8 model

Detection results	Remarks
	Detection results using the 200-epoch model. The model successfully detects a single script character BENG with a confidence of 97%.
	Detection results using the 200-epoch model. The model successfully detects the character U with a confidence of 97% and ignores other non-Toba script objects.
	Detection results using the 200-epoch model. The model accurately detects both characters, TI with a confidence of 93% and TO with a confidence of 91%.
MAKAN NASI	Detection results using the 200-epoch model. The model does not display bounding boxes and class names for the detected alphabet because the tested image is not Batak Toba script.
Interface Time Lapatanna : Terjemahan Interface Time Lapatanna : Terjemahan	Detection results using the 200-epoch model. The model does not display bounding boxes and class names for the detected pattern because the tested image is not Batak Toba script.

4.3. Results and discussion of script detection with the android application

The model obtained from the previous implementation was converted from the .pt format to the tflite format. This model was then integrated into an Android application to perform real-time object detection and image uploads. During testing, the bounding boxes produced by the model were consistently accurate in locating the desired script positions. The detected class names matched the annotated scripts, indicating that the model has good classification capability. The detection process time ranged from 1 to 2 seconds per image, depending on the image complexity (number and type of script formats).

The detection results are displayed as annotations on the camera screen. Real-time detection results are shown in Figure 11. Figure 11(a) illustrates the detection of the Batak Toba script in digital format with accuracy, detecting "BA" with a confidence of 81% and identifying the combination of "BENG" and "TU" with confidence values of 97% and 93%, respectively. Figure 11(b) shows the detection of the Batak Toba script in handwritten format, accurately identifying the combinations "MI," "TI," "TO," and "TU.".

Batak Toba script can also be detected using the image upload function on the phone, as shown in Figure 12. Users can also select images from their phone's gallery, and the application will process these images to detect and recognize the Toba script as shown in Figure 12(a) shows the detection of “RU” and “RU NYU” as for Figure 12(b) shows the detection of “DO” and “DO MA.”

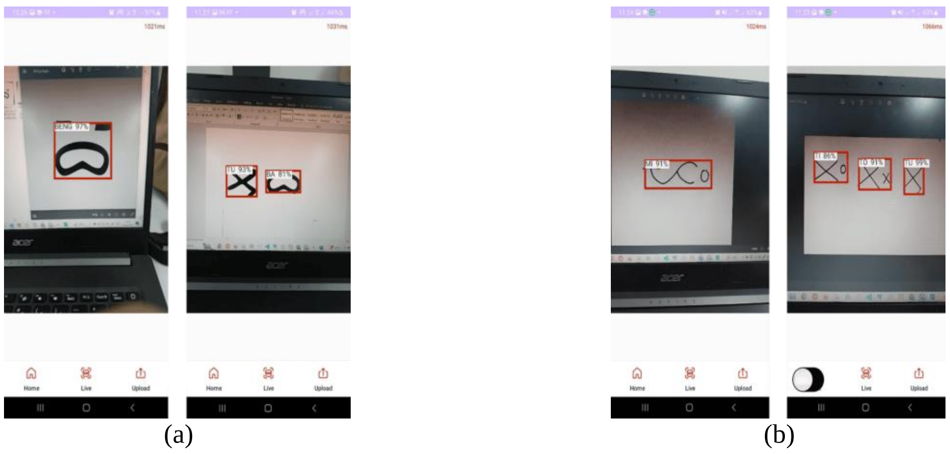


Figure 11. Detection of (a) digital script format and (b) handwritten script format

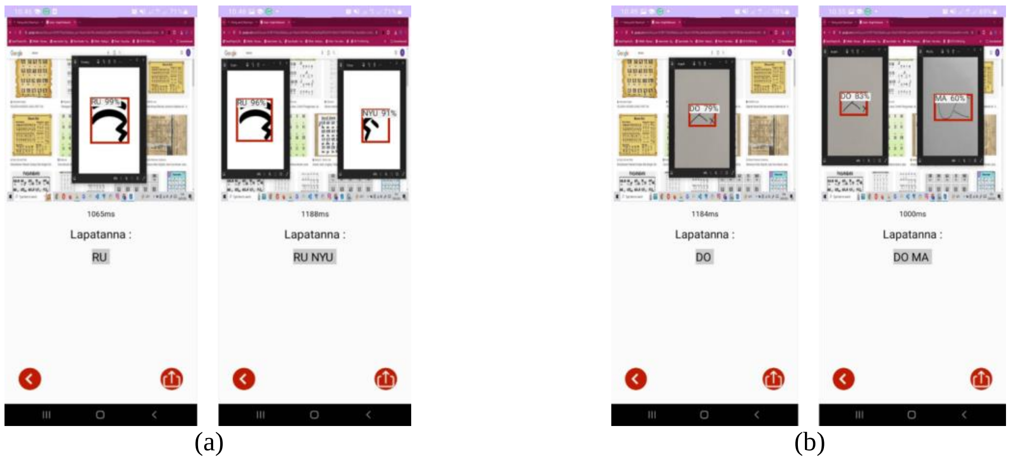


Figure 12. Detection results with uploaded (a) digital image and (b) handwritten format

5. CONCLUSION

This study has led to several key conclusions. The YOLOv8 algorithm has been successfully applied to accurately detect individual script characters and combinations of characters in the Batak Toba script. The algorithm demonstrated good confidence levels in detecting Batak Toba script objects, with hyperparameters including 200 epochs, an IoU of 0.7, and a confidence threshold of 0.5. Model evaluation yielded a precision of 0.945, recall of 0.902, and mAP of 0.954. Additionally, the Batak Toba script object detection results can be integrated into an Android mobile application using TensorFlow Lite. The application successfully detects Batak Toba script in real-time and from uploaded images, displays class labels, and reads scripts in word form, facilitating user recognition of Batak Toba script meanings.

To enhance future research, it is advisable to collect various handwritten samples of the Batak Toba script, expand the diversity of data sources, and employ different preprocessing techniques to boost model accuracy and quality. Additionally, utilizing advanced GPU hardware will facilitate training with more complex parameter configurations, speeding up the process and improving overall model performance. A continued focus on real-world testing, user feedback, and iterative refinements will be crucial to maximizing the application's potential and usability.

FUNDING INFORMATION

This research was funded by the Institute for Research and Community Service (LPPM), Institut Teknologi Del through an internal research grant scheme. The authors gratefully acknowledge the financial support provided by LPPM IT Del.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Iustisia Natalia Simbolon	✓	✓		✓						✓		✓		
Herimanto												✓		
Ranty Deviana Siahaan												✓		
Samuel Adika Lumbantobing		✓	✓	✓	✓			✓	✓					
Grace Natalia Br Sitepu		✓	✓	✓	✓			✓	✓					

C	:	**C**onceptualization	I	:	**I**nvestigation	Vi	:	**Vi**sualization
M	:	**M**ethodology	R	:	**R**esources	Su	:	**Su**pervision
So	:	**So**ftware	D	:	**D**ata Curation	P	:	**P**roject administration
Va	:	**Va**lidation	O	:	Writing - **O**riginal Draft	Fu	:	**Fu**nding acquisition
Fo	:	**Fo**rmal analysis	E	:	Writing - Review & **E**diting			

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

INFORMED CONSENT

We have obtained informed consent from all individuals included in this study.

ETHICAL APPROVAL

The research related to human use has been complied with all the relevant national regulations and institutional policies in accordance with the tenets of the Helsinki Declaration and has been approved by the authors' institutional review board or equivalent committee.

DATA AVAILABILITY

The data that support the findings of this study are openly available in Google Drive at https://bit.ly/4eSlU3n.

REFERENCES

[1] S. Willian, T. H. Rochadiani, and T. Sofian, "Design of Batak Toba script recognition system using convolutional neural network algorithm," *Sinkron*, vol. 8, no. 3, pp. 1609–1618, Jul. 2023, doi: 10.33395/sinkron.v8i3.12617.

[2] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, vol. 1, 2018.

[3] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[4] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2020, pp. 10778–10787, doi: 10.1109/CVPR42600.2020.01079.

[5] M. Fiorucci, M. Khoroshiltseva, M. Pontil, A. Traviglia, A. Del Bue, and S. James, "Machine learning for cultural heritage: a survey," *Pattern Recognition Letters*, vol. 133, pp. 102–108, May 2020, doi: 10.1016/j.patrec.2020.02.017.

[6] D. F. Lubis and T. A. Bowo, "Batak Toba script, preserving its authenticity in globalization stream," *Nusantara Science and Technology Proceedings*, pp. 28–34, Jan. 2022, doi: 10.11594/nstp.2021.1704.

[7] 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, 2021, doi: 10.1016/j.procs.2022.01.135.

[8] C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao, "YOLOv7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2023, vol. 2023-June, pp. 7464–7475, doi: 10.1109/CVPR52729.2023.00721.

[9] X. Liu, Y. Wang, D. Yu, and Z. Yuan, "YOLOv8-FDD: A real-time vehicle detection method based on improved YOLOv8," *IEEE Access*, vol. 12, pp. 136280–136296, 2024, doi: 10.1109/ACCESS.2024.3453298.

[10] Y. Pratama, S. T. N. Nainggolan, D. I. Nadya, and N. Y. Naipospos, "One-shot learning Batak Toba character recognition using siamese neural network," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 21, no. 3, pp. 600–612, Jun. 2023, doi: 10.12928/TELKOMNIKA.v21i3.24927.

[11] G. Shu, W. Liu, X. Zheng, and J. Li, "IF-CNN: image-aware inference framework for CNN with the collaboration of mobile devices and cloud," *IEEE Access*, vol. 6, pp. 68621–68633, 2018, doi: 10.1109/ACCESS.2018.2880196.

[12] C. Chen *et al.*, “YOLO-based UAV technology: a review of the research and its applications,” *Drones*, vol. 7, no. 3, p. 190, Mar. 2023, doi: 10.3390/drones7030190.
[13] S. Zhou *et al.*, “An accurate detection model of takifugu rubripes using an improved YOLO-V7 network,” *Journal of Marine Science and Engineering*, vol. 11, no. 5, p. 1051, May 2023, doi: 10.3390/jmse11051051.
[14] H. Huang and K. Zhu, “Automotive parts defect detection based on YOLOv7,” *Electronics (Switzerland)*, vol. 13, no. 10, p. 1817, May 2024, doi: 10.3390/electronics13101817.
[15] X. Wang, X. Lin, and X. Dang, “Supervised learning in spiking neural networks: A review of algorithms and evaluations,” *Neural Networks*, vol. 125, pp. 258–280, May 2020, doi: 10.1016/j.neunet.2020.02.011.
[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2016, vol. 2016-December, pp. 779–788, doi: 10.1109/CVPR.2016.91.
[17] Q. C. Mao, H. M. Sun, Y. B. Liu, and R. S. Jia, “Mini-YOLOv3: real-time object detector for embedded applications,” *IEEE Access*, vol. 7, pp. 133529–133538, 2019, doi: 10.1109/ACCESS.2019.2941547.
[18] T. Sathies Kumar and T. Sheela, “Detection of tumor in ultrasound images,” *International Journal of Recent Technology and Engineering*, vol. 8, no. 2, pp. 5288–5290, Jul. 2019, doi: 10.35940/ijrte.B1063.078219.
[19] Z. Zhang, J. Huang, G. Hei, and W. Wang, “YOLO-IR-Free: an improved algorithm for real-time detection of vehicles in infrared images,” *Sensors (Basel, Switzerland)*, vol. 23, no. 21, p. 8723, Oct. 2023, doi: 10.3390/s23218723.
[20] J. Shen, Z. Zhang, J. Luo, and X. Zhang, “YOLOv5-TS: Detecting traffic signs in real-time,” *Frontiers in Physics*, vol. 11, Nov. 2023, doi: 10.3389/fphy.2023.1297828.
[21] H. Lou *et al.*, “DC-YOLOv8: small-size object detection algorithm based on camera sensor,” *Electronics (Switzerland)*, vol. 12, no. 10, p. 2323, May 2023, doi: 10.3390/electronics12102323.
[22] Y. C. How, A. F. Ab. Nasir, K. F. Muhammad, A. P. P. Abdul Majeed, M. A. M. Razman, and M. A. Zakaria, “Glove defect detection via YOLO V5,” *Mekatronika*, vol. 3, no. 2, pp. 25–30, Jan. 2022, doi: 10.15282/mekatronika.v3i2.7342.
[23] K. Dokic, M. Martinovic, and D. Mandusic, “Inference speed and quantisation of neural networks with TensorFlow Lite for Microcontrollers framework,” in *SEEDA-CECNSM 2020 - 5th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference*, Sep. 2020, pp. 1–6, doi: 10.1109/SEEDA-CECNSM49515.2020.9221846.
[24] M. Abadi, “TensorFlow: learning functions at scale,” *ACM SIGPLAN Notices*, vol. 51, no. 9, pp. 1–1, Sep. 2016, doi: 10.1145/3022670.2976746.
[25] R. S. Praneeth, K. C. S. Akash, B. K. Sree, P. I. Rani, and A. Bhola, “Scaling object detection to the edge with YOLOv4, TensorFlow lite,” in *Proceedings - 7th International Conference on Computing Methodologies and Communication, ICCMC 2023*, Feb. 2023, pp. 1547–1552, doi: 10.1109/ICCMC56507.2023.10084319.

BIOGRAPHIES OF AUTHORS

Iustisia Natalia Simbolon is current lecturer and researcher member in Informatics Department at Institut Teknologi Del since 2019. Have experience specializing in computer vision, machine learning, UI/UX design, database system, cryptography and information security, and software engineering. She dedicates himself to university teaching and conducting research. Her research interests include computer vision, machine learning, software engineering, UI/UX design. She can be contacted at email: iustisia.simbolon@del.ac.id or iustisians@gmail.com.

Herimanto received his S.Kom. (bachelor of computer science) degree in computer science from Universitas Sumatera Utara in 2017 and his M.Kom. (master of computer science) degrees in informatics from Universitas Utara in 2021. His academic pursuits focused on computer science and artificial intelligence, particularly object detection, computer vision, machine learning, and deep learning. He currently contributes his expertise as a lecturer and researcher at the Institut Teknologi Del, Toba, Indonesia. He can be contacted at email: herimanto.pardede@del.ac.id.

Ranty Deviana Siahaan received her B.I.E. (bachelor of informatics engineering) from Institut Teknologi Del in 2018 and her M.Eng. (master of engineering) from Gadjah Mada University in 2023. Her academic pursuits focused on human-centered software engineering, software quality assurance, artificial intelligence, and informatics in education. She currently contributes her expertise as a lecturer and researcher at the Institut Teknologi Del, Toba, Indonesia. She can be contacted at email: ranty.siahaan@del.ac.id.

Samuel Adika Lumbantobing is an informatics student at Institut Teknologi Del, committed to continuous learning and creative problem-solving with a specialization in software engineering and backend development. His expertise includes mobile and web application development, with experience working at BTPN Syariah, Bold & Underline, and Matakail Communication. Samuel is actively involved in innovative projects such as a facial emotion recognition application using deep learning techniques with an accuracy of 83%, a website enabling users to manage items, particularly PlayStation CDs from generations 1 to 5, a scanner application for the FUNtasTRI event, and a mobile app for real-time detection of Batak characters using the YOLOv8 algorithm. He has received various awards, including being a finalist in the WIKIDATA competition and winning first place. Samuel has also served as a project leader, product manager, full-stack developer, and tester in various technology projects. His research interests include software development, backend engineering, and real-time applications. He can be contacted via email: samueltobing37@gmail.com.

Grace Natalia Br Sitepu is an undergraduate student at Institut Teknologi Del, majoring in informatics. She has experience working on UI/UX and web development projects. Her notable projects include the SINOW web application, where she worked as a front-end developer using ReactJS, and the EBus web application, an online ticket purchasing platform developed with Spring Boot and Angular CLI. Grace has also contributed to user interface development for the scholarship application mobile project and served as the team leader for UI/UX teams in projects such as Funtastri: Check-In Barcode Mobile Web UI and the Redesign of Simbisa Tourist Village Website. Additionally, she designed the interface for Geting Mobile App, a delivery service application. Grace’s research interests focus on user interface development and the creation of responsive and user-friendly web applications. She can be contacted via email: sitepu12grace@gmail.com.