JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER

p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 1 June 2026
Hal: 421-429
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i1.1039

Leakage-Aware Benchmarking of Lightweight Models for Robust Handwritten Hanacaraka Character Recognition

Ferian Fauzi Abdulloh[1*], Sharazita Dyah Anggita[2], Ikmah[3], Ali Mustopa[4], Majid Rahardi[5], Devi Wulandari[6]

[1,5]Faculty of Computer Science, Informatics Study Program, AMIKOM University, Sleman, Indonesia
[2,3,4]Faculty of Computer Science, Information System Study Program, AMIKOM University, Sleman, Indonesia
Email: ferian@amikom.ac.id[1], sharazita@amikom.ac.id[2], ikmahdarwan01@amikom.ac.id[3], ali.m@amikom.ac.id[4], majid@amikom.ac.id[5], devi@amikom.ac.id[6]
(*Email Corresponding Author: ferian@amikom.ac.id)
Received: May 10, 2026 | Revision: May 18, 2026 | Accepted: May 19, 2026

Abstract

Handwritten Hanacaraka character recognition is important for preserving Javanese script, but reported results can be difficult to compare because datasets, preprocessing procedures, and train-test separation protocols vary across studies. This study presents a leakage-aware benchmark of lightweight models on a public handwritten Hanacaraka dataset containing 20 basic character classes. A data audit removed 17 unreadable image files and retained 1,562 valid images. Two experimental settings were evaluated: a perceptual-hash grouped split for leakage-aware testing and a random-stratified split as an optimistic upper-bound scenario. The leakage-aware benchmark compared HOG with SVM, HOG with Random Forest, MobileNetV2 head-only training, fine-tuned MobileNetV2, and a confusion-aware MobileNetV2 variant. Fine-tuned MobileNetV2 achieved the best leakage-aware result with 53.82% accuracy and 49.59% macro-F1, while robustness testing under image distortions produced 47.85% accuracy and 44.53% macro-F1. In the optimistic random-stratified experiment, an ensemble of EfficientNetB0 and MobileNetV2 with test-time augmentation reached 74.11% accuracy and 74.24% macro-F1. The results indicate that stricter evaluation substantially lowers performance and that visually similar classes remain difficult. Therefore, future Hanacaraka recognition work should report leakage control, robustness, and confusion analysis, not only clean-set accuracy.

Keywords : Hanacaraka, Javanese script, handwritten recognition, leakage-aware evaluation, lightweight models

1. INTRODUCTION

Javanese script, commonly known as Hanacaraka or Aksara Jawa, is one of Indonesia's important traditional writing systems. Although it is taught as cultural and linguistic heritage, its everyday usage has decreased, and many learners find the shapes of the characters difficult to recognize consistently. Digital recognition systems can support learning media, transliteration tools, and preservation-oriented applications by converting handwritten or captured script images into machine-readable labels. However, handwritten Hanacaraka recognition is not a trivial image classification task. The character set contains visually related strokes, several classes have similar global silhouettes, and handwritten samples introduce variation in stroke thickness, orientation, proportion, and personal writing style. These conditions make a model appear accurate on clean data but fail when the evaluation split is stricter or when the input is distorted.

Previous studies show that Javanese character recognition has been approached using both classical machine learning and deep learning. Dewa et al. compared CNN and DNN-style neural approaches for handwritten Javanese recognition and reported that CNN was better suited for spatial character images [1]. Rasyidi et al. used preprocessing and Histogram of Oriented Gradients (HOG) with Random Forest and reported high performance for 20 handwritten Javanese classes [2]. Anggraeny et al. emphasized that preprocessing choices such as binarization, thinning, and denoising influence handwritten Javanese character recognition [3]. More recent deep-learning studies have explored custom CNN architectures, twelve-layer DCNN designs with augmentation, DenseNet, ResNet-18, and transfer-learning pipelines [4]-[8]. These studies are useful because they demonstrate the feasibility of learning discriminative visual features for Hanacaraka, but their experimental protocols are not always directly comparable.

A common pattern in the literature is the reporting of very high clean-set accuracy. For example, custom CNN and DenseNet-based studies have reported results above 97% or even near-perfect accuracy under their own datasets and protocols [5], [6]. Another line of work applied transfer learning for Javanese script transliteration and compared architectures such as MobileNetV2, VGG, and Xception [8]. Mobile-client recognition systems have also been proposed to bring Javanese letter recognition into mobile learning environments [9]. These works confirm that deep convolutional features are useful; however, they also raise an important question: whether high accuracy remains stable after duplicate-like samples are controlled, corrupt files are removed, and robustness to image distortion is measured.

This issue is especially important for small public datasets. When a dataset contains visually similar or near-duplicate samples, a random split can place almost identical images in both training and testing partitions. In that situation, the model may learn dataset-specific appearance rather than class-generalizable structure. The problem is not unique to Hanacaraka; it is a general concern in small-scale computer vision benchmarking. However, it is particularly relevant here because the public Hanacaraka dataset used in this study contains a limited number of images per class and includes files

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 1 June 2026
Hal: 421-429
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i1.1039

that appear to have unreadable image content. Therefore, before model comparison, the dataset must be audited, invalid samples must be removed, and the split protocol must be explicitly stated.

Another gap is that many studies emphasize clean accuracy but provide less detail about failure modes. For a character recognition system, a single overall accuracy value hides whether the system performs equally across all classes. In practice, a learning application or OCR tool may be less useful if it repeatedly confuses specific pairs such as ba and nya, na and ka, ha and ya, or la and ya. Confusion analysis is therefore necessary to identify the unresolved visual relationships between character classes. Robustness evaluation is also important because real user input may be blurred, rotated, noisy, or captured under different brightness conditions.

This study addresses these gaps by benchmarking lightweight models for handwritten Hanacaraka recognition under a stricter evaluation workflow. The contributions are fourfold. First, a data audit is performed to remove unreadable files and report the real number of valid images. Second, a leakage-aware split is created using perceptual-hash grouping to reduce near-duplicate leakage between training, validation, and test sets. Third, lightweight classical and deep-learning models are compared, including HOG with SVM, HOG with Random Forest, MobileNetV2, and a confusion-aware MobileNetV2 variant. Fourth, a separate random-stratified upper-bound experiment with EfficientNetB0, MobileNetV2, test-time augmentation, and ensemble averaging is reported to show the performance difference between optimistic and stricter evaluation scenarios. The goal is not to claim a new state-of-the-art model, but to provide a transparent benchmark and identify where Hanacaraka recognition remains difficult.

2. RESEARCH METHODOLOGY

2.1 Research Stages

The research process consisted of dataset acquisition, image validity checking, preprocessing, split construction, model training, clean-set testing, robustness testing, and interpretation of confusion patterns. The workflow is shown in Figure 1. The dataset was first scanned using the Python Imaging Library to verify that each file could be opened and decoded. Invalid images were excluded from the dataframe but were not manually corrected, so the benchmark reflects a reproducible cleaning procedure. The valid images were then resized according to each model configuration. HOG-based baselines used grayscale features, while deep-learning models used RGB tensors with architecture-specific preprocessing.

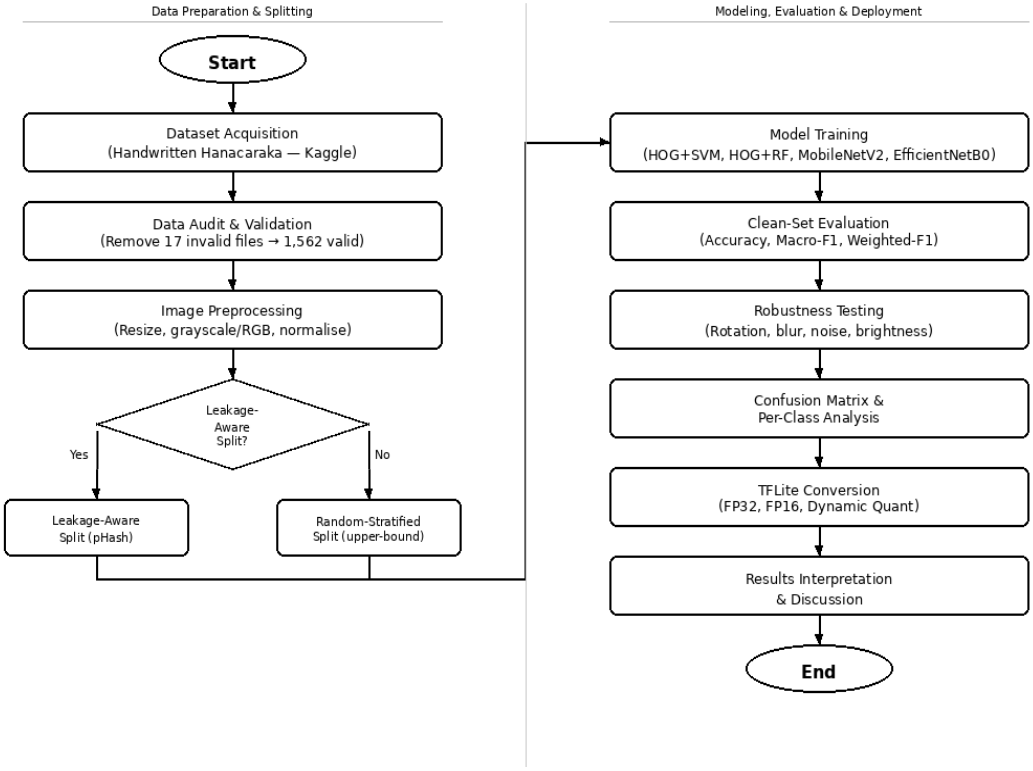

Figure 1. Research process flow for leakage-aware Hanacaraka benchmarking

2.2 Dataset Audit and Splitting Strategy

The experiment used a public handwritten Hanacaraka dataset collected from digital handwriting forms and scanned handwritten sources [18]. The dataset represents the 20 basic classes: ba, ca, da, dha, ga, ha, ja, ka, la, ma, na, nga, nya, pa, ra, sa, ta, tha, wa, and ya. After the image validation process, 1,562 images were retained and 17 unreadable files were

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 1 June 2026
Hal: 421-429
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i1.1039

removed. The retained dataset was nearly balanced, with most classes containing between 75 and 79 valid images. The class distribution is summarized in Table 1.

Table 1. Valid image distribution after audit

Class	Count	Class	Count	Class	Count	Class	Count
ba	79	ca	79	da	79	dha	77
ga	78	ha	78	ja	78	ka	79
la	78	ma	79	na	79	nga	79
nya	78	pa	77	ra	75	sa	79
ta	76	tha	77	wa	79	ya	79

The main benchmark used a leakage-aware split. Perceptual hashing was applied to group visually similar images so that images with identical or highly similar perceptual signatures were not intentionally separated across training and testing partitions. The resulting leakage-aware split contained 998 training images, 250 validation images, and 314 test images. This split was used for the primary scientific conclusion. A secondary random-stratified split was also used in the accuracy-search experiment, producing 999 training images, 281 validation images, and 282 test images. The random-stratified setting was treated as an optimistic upper-bound scenario, not as the main leakage-aware evidence.

Table 2. Experimental split configurations

Split mode	Train	Validation	Test	Purpose
Leakage-aware pHash grouped	998	250	314	Primary benchmark
Random stratified	999	281	282	Upper-bound experiment

2.3 Models and Training Configuration

The benchmark compared classical machine-learning baselines and lightweight transfer-learning models. The classical baselines used HOG descriptors [14], followed by Support Vector Machine [15] and Random Forest [16] classifiers implemented using scikit-learn [17]. These baselines were included because handcrafted gradient descriptors and tree/kernel classifiers remain common comparisons in handwritten character recognition. The deep-learning baseline used MobileNetV2, a compact architecture based on inverted residual blocks and linear bottlenecks [11]. MobileNetV2 was trained in two stages: a head-only phase with the pretrained backbone frozen, followed by a fine-tuning phase in which selected upper layers were unfrozen while Batch Normalization layers remained frozen for stability. A confusion-aware variant was also tested by increasing the emphasis on classes involved in frequent validation confusions and by using a class-balanced focal-loss formulation inspired by Lin et al. [13].

The accuracy-search experiment extended the candidate models to EfficientNetB0, which uses compound scaling to balance network depth, width, and input resolution [12]. The search evaluated MobileNetV2 and EfficientNetB0 at different input sizes, cross-entropy and focal-loss objectives, fine-tuning, test-time augmentation (TTA), and ensemble averaging. TTA averaged predictions over multiple image variants at inference time. The ensemble prediction was obtained by averaging class-probability vectors from the top individual models and selecting the class with the highest averaged probability.

Table 3. Compared model families

Model	Input feature	Classifier / architecture	Role
HOG + SVM	Grayscale HOG	Support Vector Machine	Classical baseline
HOG + Random Forest	Grayscale HOG	Random Forest	Tree-ensemble baseline
MobileNetV2 HeadOnly	RGB image	Frozen MobileNetV2 + dense head	Lightweight transfer baseline
MobileNetV2 FineTune	RGB image	Partial fine-tuning	Primary deep-learning model
ConfusionAware MobileNetV2	RGB image	Class-balanced focal training	Hard-class training variant
EfficientNetB0 / Ensemble	RGB image	Fine-tuning, TTA, probability averaging	Upper-bound accuracy search

2.4 Evaluation Metrics and Robustness Testing

Model performance was evaluated using accuracy, macro-F1, and weighted-F1. Accuracy measures the proportion of correct predictions, while macro-F1 averages F1-scores equally across the 20 classes and is therefore sensitive to weak minority or difficult classes. Weighted-F1 accounts for support size. Because the class distribution was nearly balanced, accuracy and weighted-F1 were expected to be similar, but macro-F1 was prioritized for interpreting class-level weaknesses. Robustness testing was conducted by applying image distortions such as rotation, blur, brightness

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 1 June 2026
Hal: 421-429
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i1.1039

changes, noise, and stroke-like perturbations to test images. The goal was to evaluate whether the best clean-set model remained reliable under input conditions that are closer to practical use.

For mobile deployment feasibility, the best MobileNetV2 model was exported to TensorFlow Lite in FP32, FP16, and dynamic-range quantized formats. The exported file sizes were recorded to assess whether the model family could support lightweight deployment on devices with limited storage and computation. The present study did not perform direct Android latency measurement; therefore, deployment discussion is limited to conversion success and model size.

3. RESULTS AND DISCUSSION

3.1 Dataset Audit Results

The audit stage found 17 files that had image extensions but could not be decoded by the image loader. Removing these files prevented training and preprocessing failures and produced a clean working set of 1,562 valid images. This step is important because public datasets may contain broken or incorrectly encoded files, and ignoring them can cause notebooks to fail or silently produce inconsistent data counts. The final class distribution remained balanced, with the smallest classes containing 75 valid images and the largest classes containing 79 valid images. Therefore, the main difficulty in this dataset is not severe class imbalance, but limited total sample size and visual similarity among classes.

The use of perceptual-hash grouping produced a more conservative split than ordinary random stratification. This choice intentionally makes the task harder because near-duplicate images are less likely to be shared across training and testing partitions. In a small handwritten dataset, this is necessary to avoid overestimating generalization. The difference between the leakage-aware benchmark and the random-stratified upper-bound experiment later confirms that split design can strongly affect reported performance.

3.2 Leakage-Aware Benchmark Performance

Table 4 presents the primary leakage-aware benchmark. Fine-tuned MobileNetV2 achieved the best result, with 53.82% accuracy, 49.59% macro-F1, and 50.69% weighted-F1. The confusion-aware MobileNetV2 variant was very close, achieving 53.18% accuracy and 49.53% macro-F1, but it did not surpass the standard fine-tuned MobileNetV2. MobileNetV2 head-only training achieved 51.91% accuracy, while HOG with Random Forest and HOG with SVM achieved 44.27% and 37.26% accuracy, respectively. These results show that transfer learning from a lightweight CNN provides a clear advantage over handcrafted HOG baselines, but the advantage is not sufficient to make the task easy under leakage-aware evaluation.

Table 4. Leakage-aware benchmark results on the test set

Model	Accuracy	Macro-F1	Weighted-F1
MobileNetV2 FineTune	53.82%	49.59%	50.69%
ConfusionAware MobileNetV2	53.18%	49.53%	50.50%
MobileNetV2 HeadOnly	51.91%	48.51%	49.72%
HOG RandomForest	44.27%	42.97%	43.39%
HOG SVM	37.26%	37.37%	38.06%

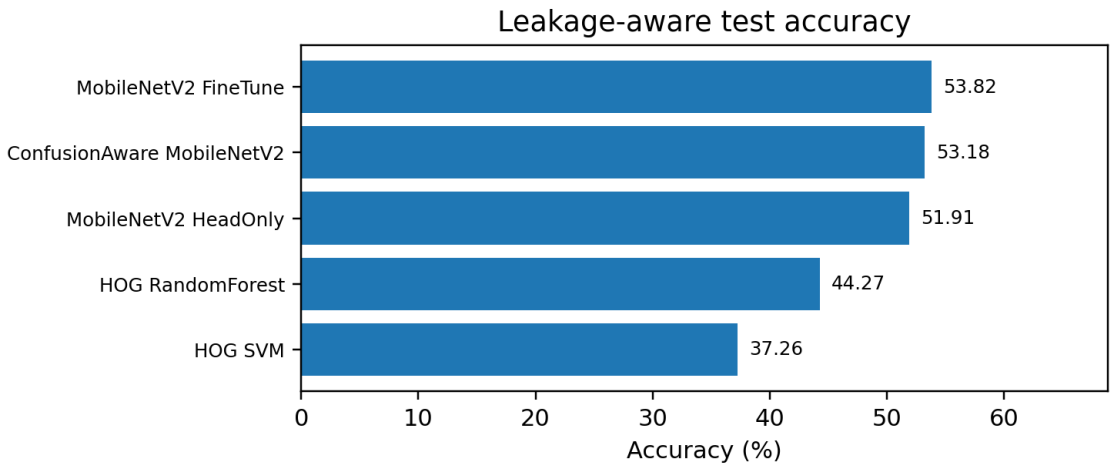

Figure 2. Leakage-aware test accuracy comparison

The relatively modest result is an important finding. It differs from some high-accuracy reports in previous Hanacaraka or Javanese script studies [4]-[7]. The difference should not be interpreted as a contradiction, because datasets and evaluation protocols are not identical. Instead, it shows that high clean-set accuracy is not enough evidence of robust recognition when near-duplicate leakage, corrupt files, and realistic distortions are considered. The best leakage-aware

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 1 June 2026
Hal: 421-429
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i1.1039

model correctly classified only slightly more than half of the test images, despite using ImageNet-pretrained features. This suggests that the public dataset is not yet large enough to support robust 20-class generalization under a strict split, especially when characters have similar stroke arrangements.

The confusion-aware variant was designed to improve hard-class performance by emphasizing classes mined from validation confusions. Its near-equal macro-F1 indicates that the idea was not harmful, but it also did not provide a measurable improvement over ordinary fine-tuning. There are two possible explanations. First, class-level weighting may be too coarse because confusion is pair-specific: increasing the weight of a class does not directly teach the model which visually similar alternative should be separated from it. Second, the model may need more diverse examples of the hard pairs rather than higher loss weight on the same limited samples. Future work should test pairwise contrastive learning, writer-independent data collection, or synthetic character generation targeted at the most confused pairs.

3.3 Robustness Evaluation

Table 5 summarizes robustness testing for the three MobileNetV2-based models. Fine-tuned MobileNetV2 remained the best model under distortions, with 47.85% accuracy and 44.53% macro-F1. Confusion-aware MobileNetV2 achieved 47.13% accuracy and 44.16% macro-F1, while MobileNetV2 HeadOnly achieved 45.82% accuracy and 42.82% macro-F1. The decrease from clean accuracy to robustness accuracy confirms that handwritten Hanacaraka recognition is sensitive to visual perturbations such as rotation, blur, brightness variation, and noise. This is expected because character recognition depends on local stroke geometry, and small distortions can obscure the features that differentiate similar glyphs.

Table 5. Robustness evaluation under image distortions

Model	Robust Accuracy	Robust Macro-F1	Robust Weighted-F1
MobileNetV2 FineTune	47.85%	44.53%	45.34%
ConfusionAware MobileNetV2	47.13%	44.16%	45.05%
MobileNetV2 HeadOnly	45.82%	42.82%	43.65%

The robustness gap is important for mobile learning applications. A user may write a character on paper, capture it using a camera, or submit a digital drawing with uneven stroke width. If the evaluation only uses clean cropped images, the system may look promising but underperform in practice. Therefore, robustness metrics should be reported alongside clean accuracy in future work. The present results also justify the paper title's emphasis on robustness evaluation: the benchmark is not merely a competition between architectures, but an assessment of practical reliability.

3.4 Confusion and Per-Class Analysis

The confusion matrix revealed that errors were not uniformly distributed. Table 6 lists the largest confusion pairs from the best leakage-aware model. The most frequent error was ba predicted as nya, followed by la predicted as ya, na predicted as ka, ga predicted as ya, ha predicted as ya, and la predicted as ha. These patterns show that the main unresolved issue is hard-character separation. The model often predicts a visually related class rather than a random class, meaning the learned representation captures some global structure but fails to distinguish fine stroke differences.

Table 6. Top confusion pairs for MobileNetV2 FineTune

True class	Predicted class	Error count
ba	nya	8
la	ya	7
na	ka	6
ga	ya	6
ha	ya	6
la	ha	5
ca	ka	4
ha	ka	4
sa	pa	4
ra	ja	4

Table 7. Weakest and strongest classes in the leakage-aware test set

Weak class	F1-score	Strong class	F1-score
la	0.00%	ja	87.80%
ba	17.39%	ra	69.57%
ha	21.43%	ga	66.67%
ca	30.77%	tha	66.67%
ta	37.50%	wa	64.00%
na	40.00%	ma	63.83%
sa	44.44%	ka	63.64%

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 1 June 2026
Hal: 421-429
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i1.1039

The weakest class was la, with an F1-score of 0.00% in this split, followed by ba, ha, ca, ta, na, and sa. The strongest classes included ja, ra, ga, tha, wa, ma, and ka. The low score for la should not be interpreted as a universal property of the letter; rather, it reflects the interaction between the split, the limited number of examples, and the similarity of la to several other classes in the test partition. Still, the result is useful because it identifies which labels require closer attention in additional data collection or augmentation. A balanced dataset by class count does not guarantee balanced difficulty by visual separability.

3.5 Optimistic Accuracy-Search Experiment

Because the user-oriented goal of the experimental campaign was to push accuracy as high as possible, an additional random-stratified upper-bound experiment was performed. This experiment used stronger configurations, including EfficientNetB0, MobileNetV2 at different input sizes, focal loss, fine-tuning, TTA, and ensemble averaging. The best single model was EfficientNetB0 with 224-pixel input, focal loss, fine-tuning, and TTA, which achieved 70.92% accuracy and 70.41% macro-F1. The best overall model was Ensemble_Top2, combining the best EfficientNetB0 and MobileNetV2 TTA models by probability averaging, with 74.11% accuracy and 74.24% macro-F1. These results are shown in Table 8 and Figure 3.

Table 8. Best random-stratified upper-bound results

Model	Accuracy	Macro-F1	Weighted-F1	Type
Ensemble Top2	74.11%	74.24%	74.13%	Ensemble
Ensemble Top4	74.11%	73.96%	73.88%	Ensemble
Ensemble Top3	73.76%	73.70%	73.64%	Ensemble
EFFB0 224 FOCAL FineTune TTA	70.92%	70.41%	70.23%	Single model
MNV2 160 CE FineTune TTA	69.86%	69.52%	69.50%	Single model
EFFB0 192 CE FineTune TTA	69.50%	69.12%	69.08%	Single model
EFFB0 224 FOCAL FineTune	68.44%	67.70%	67.55%	Single model
EFFB0 224 FOCAL Head TTA	68.09%	66.59%	66.49%	Single model

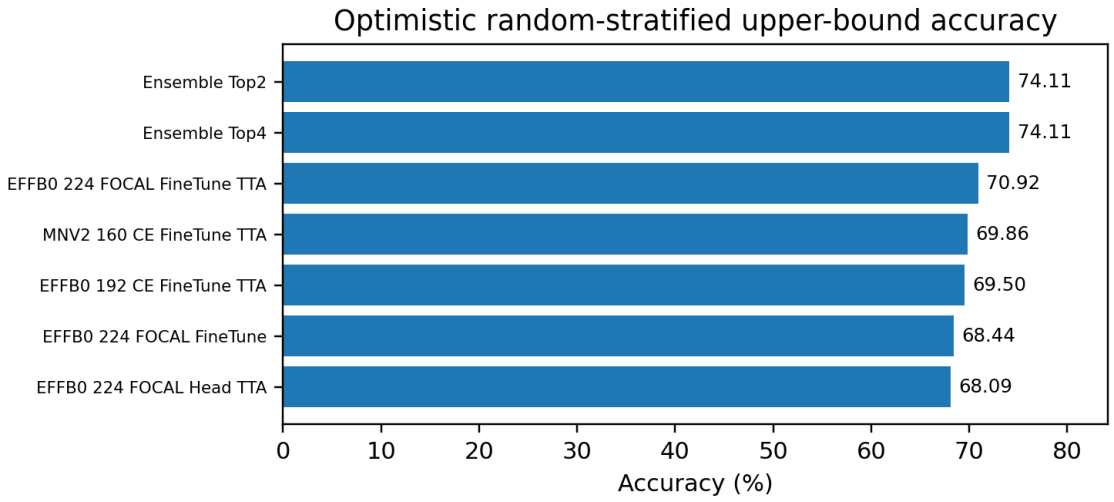

Figure 3. Upper-bound accuracy-search results under random-stratified split

Although the upper-bound experiment improved accuracy substantially, it must be interpreted carefully. The random-stratified split is more optimistic because it does not explicitly group near-duplicate images by perceptual hash. Therefore, the 74.11% ensemble result is useful as an engineering target but should not replace the leakage-aware benchmark as the main scientific evidence. The difference between 53.82% in the leakage-aware setting and 74.11% in the random-stratified setting supports the central argument of this paper: split protocol strongly influences reported performance in low-resource Hanacaraka recognition. Researchers should state whether their reported accuracy is leakage-aware, random-stratified, augmented, or evaluated on a separate writer-independent dataset.

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 1 June 2026
Hal: 421-429
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i1.1039

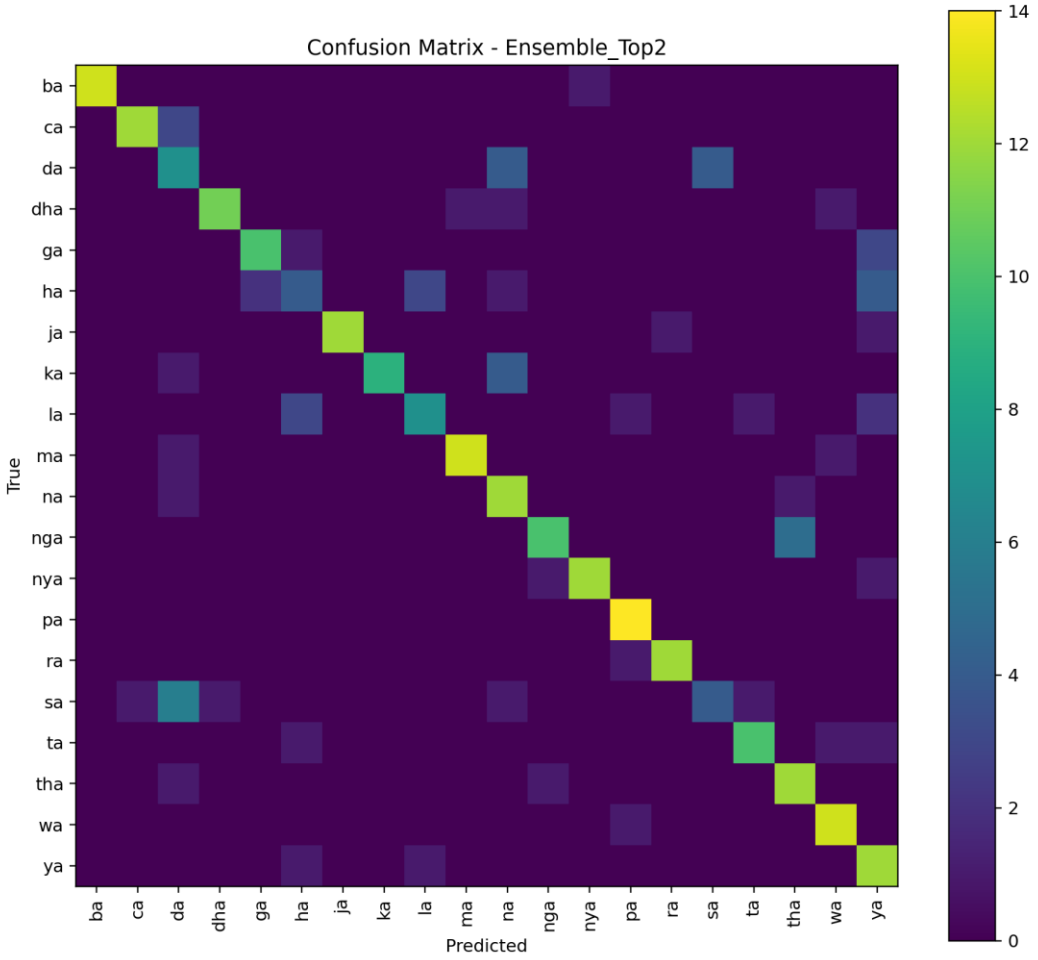

Figure 4. Confusion matrix of the best random-stratified Ensemble_Top2 model

The best Ensemble_Top2 model still exhibited difficult pairs. In the random-stratified confusion matrix, sa was frequently predicted as da, nga was predicted as tha, da was predicted as sa or na, ha was confused with ga, la, and ya, and ka was confused with na. Therefore, even when the accuracy-search setting improves the score, the same fundamental problem remains: several Hanacaraka characters are difficult to separate from image appearance alone when the available data are limited. A future model may require character-structure priors, pairwise metric learning, writer-aware sampling, or generation of controlled synthetic variants for specific hard pairs.

3.6 TFLite Conversion and Practical Deployment

For practical applications, lightweight deployment is important because Hanacaraka learning tools may run on mobile devices. The best leakage-aware MobileNetV2 model was exported to TensorFlow Lite. The FP32 model size was 9.72 MB, the FP16 model size was 4.89 MB, and dynamic-range quantization reduced the model to 2.71 MB. These file sizes indicate that MobileNetV2 is feasible for mobile storage constraints. However, this study did not measure latency on real Android devices, so runtime claims should be limited. Future work should report inference time, memory usage, battery impact, and accuracy after full integer quantization on representative mobile hardware.

Table 9. TensorFlow Lite export sizes for MobileNetV2 FineTune

Export format	Model size
fp32	9.72 MB
dynamic_quant	2.71 MB
fp16	4.89 MB

3.7 Discussion of Research Gap

The main contribution of this work is not a claim of a superior architecture, but a more cautious evaluation perspective. The experiments show that fine-tuned MobileNetV2 is the strongest tested leakage-aware model, but its accuracy remains modest. The proposed confusion-aware strategy almost matched the baseline but did not improve it, indicating that simple class reweighting is insufficient for hard-character separation. The accuracy-search experiment showed that stronger

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 1 June 2026
Hal: 421-429
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i1.1039

backbones, TTA, and ensembling can raise performance to 74.11% under an optimistic split, but this still does not reach 90% and should not be reported as leakage-aware performance.

These findings are relevant to the broader literature because some previous works report very high accuracy on Javanese script recognition [4]-[7], while older CNN/DNN recognition studies reported substantially lower values [1]. The difference suggests that dataset design, split protocol, augmentation, and task definition can be as important as model architecture. A 20-class isolated-character task with near-duplicate random splits may produce much higher numbers than a split designed to reduce visual leakage. Therefore, future studies should include dataset manifests, duplicate checks, split scripts, and robustness tests. Without these details, reported accuracies are difficult to compare fairly.

The paper also shows that lightweight models remain attractive for Hanacaraka recognition. MobileNetV2 performed consistently well across strict and robust settings, while EfficientNetB0 became the best single model in the upper-bound search. However, the unresolved error cases suggest that architecture selection alone is unlikely to solve the task. A more promising direction is targeted data improvement: collecting more samples for weak classes, ensuring writer-independent evaluation, balancing not only class counts but also handwriting styles, and generating controlled variations for specific confusion pairs. Pairwise contrastive objectives or angular-margin losses may also be useful if they are designed around empirically observed hard pairs rather than all classes equally.

Overall, the results support the title of this study. A leakage-aware benchmark changes the interpretation of model performance, and robustness evaluation reveals the practical fragility of models trained on small public datasets. The proposed paper therefore provides a transparent baseline for future Hanacaraka recognition research and argues that future claims of high accuracy should be accompanied by leakage-control evidence and hard-class analysis.

4. CONCLUSION

This study benchmarked lightweight models for handwritten Hanacaraka character recognition under leakage-aware and robustness-oriented evaluation. After auditing the public dataset, 17 invalid image files were removed and 1,562 valid images across 20 classes were retained. In the primary perceptual-hash grouped benchmark, fine-tuned MobileNetV2 achieved the best result, with 53.82% accuracy and 49.59% macro-F1, outperforming HOG with SVM, HOG with Random Forest, MobileNetV2 HeadOnly, and a confusion-aware MobileNetV2 variant. Under robustness testing, the same model remained the best but dropped to 47.85% accuracy and 44.53% macro-F1, showing that the task remains sensitive to realistic image distortions. An additional random-stratified upper-bound experiment increased the best score to 74.11% accuracy and 74.24% macro-F1 using an EfficientNetB0-MobileNetV2 ensemble with test-time augmentation, but this result should not be interpreted as leakage-aware evidence. The major unresolved problem is hard-character confusion, especially among visually similar classes such as ba-nya, la-ya, na-ka, ga-ya, ha-ya, and related pairs. Therefore, future work should prioritize writer-independent data collection, stronger duplicate-control protocols, targeted augmentation for weak classes, pairwise metric learning, and direct Android latency testing. The findings demonstrate that transparent evaluation protocol is essential for fair comparison in low-resource Hanacaraka recognition.

REFERENCES

[1] C. K. Dewa, A. L. Fadhilah, and Afiahayati, "Convolutional neural networks for handwritten Javanese character recognition," IJCCS (Indonesian Journal of Computing and Cybernetics Systems), vol. 12, no. 1, pp. 83-94, 2018, doi: 10.22146/ijccs.31144.

[2] M. A. Rasyidi, T. Bariyah, Y. I. Riskajaya, and A. D. Septyani, "Classification of handwritten Javanese script using random forest algorithm," Bulletin of Electrical Engineering and Informatics, vol. 10, no. 3, pp. 1308-1315, 2021, doi: 10.11591/eei.v10i3.3036.

[3] F. T. Anggraeny, Y. V. Via, and R. Mumpuni, "Image preprocessing analysis in handwritten Javanese character recognition," Bulletin of Electrical Engineering and Informatics, vol. 12, no. 2, pp. 860-867, 2023, doi: 10.11591/eei.v12i2.4172.

[4] A. Susanto, I. U. W. Mulyono, C. A. Sari, E. H. Rachmawanto, D. R. I. M. Setiadi, and M. K. Sarker, "Handwritten Javanese script recognition method based 12-layers deep convolutional neural network and data augmentation," IAES International Journal of Artificial Intelligence, vol. 12, no. 3, pp. 1448-1458, 2023, doi: 10.11591/ijai.v12.i3.pp1448-1458.

[5] A. Susanto, I. U. W. Mulyono, C. A. Sari, E. H. Rachmawanto, D. R. I. M. Setiadi, and M. K. Sarker, "Improved Javanese script recognition using custom model of convolution neural network," International Journal of Electrical and Computer Engineering, vol. 13, no. 6, pp. 6629-6636, 2023, doi: 10.11591/ijece.v13i6.pp6629-6636.

[6] E. D. B. Sudewo, M. K. Biddinika, and A. Fadlil, "DenseNet architecture for efficient and accurate recognition of Javanese script Hanacaraka character," MATRIK: Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer, vol. 23, no. 2, pp. 453-464, 2024, doi: 10.30812/matrik.v23i2.3855.

[7] E. D. B. Sudewo, M. K. Biddinika, and A. Fadlil, "Javanese script Hanacaraka character prediction with ResNet-18 architecture," JURTEKSI (Jurnal Teknologi dan Sistem Informasi), vol. 10, no. 2, pp. 363-370, 2024, doi: 10.33330/jurteksi.v10i2.3017.

[8] M. F. Naufal, J. Siswantoro, and J. T. Soebroto, "Transliterating Javanese script images to Roman script using convolutional neural network with transfer learning," JOIV: International Journal on Informatics Visualization, vol. 8, no. 3, pp. 1460-1468, 2024, doi: 10.62527/joiv.8.3.2566.

[9] Y. Harjoseputro, Y. D. Handarkho, and H. T. R. Adie, "The Javanese letters classifier with mobile client-server architecture and convolution neural network method," International Journal of Interactive Mobile Technologies, vol. 13, no. 12, pp. 67-80, 2019, doi: 10.3991/ijim.v13i12.11492.

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 1 June 2026
Hal: 421-429
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i1.1039

[10] A. Susanto, C. A. Sari, E. H. Rachmawanto, I. U. W. Mulyono, and N. M. Yaacob, "A comparative study of Javanese script classification with GoogleNet, DenseNet, ResNet, VGG16 and VGG19," Scientific Journal of Informatics, vol. 11, no. 1, pp. 31-40, 2024, doi: 10.15294/sji.v11i1.47305.

[11] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510-4520, doi: 10.1109/CVPR.2018.00474.

[12] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in Proc. 36th International Conference on Machine Learning, 2019, pp. 6105-6114, doi: 10.48550/arXiv.1905.11946.

[13] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in Proc. IEEE International Conference on Computer Vision, 2017, pp. 2999-3007, doi: 10.1109/ICCV.2017.324.

[14] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, pp. 886-893, doi: 10.1109/CVPR.2005.177.

[15] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, no. 3, pp. 273-297, 1995, doi: 10.1007/BF00994018.

[16] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5-32, 2001, doi: 10.1023/A:1010933404324.

[17] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011, doi: 10.5555/1953048.2078195.

[18] R. P. Nugroho, "Aksara Jawa / Hanacaraka Dataset," Kaggle, accessed May 13, 2026. [Online]. Available: https://www.kaggle.com/datasets/vzrenggamani/hanacaraka. DOI: N/A (online dataset).