Journal of Deep Learning, Computer Vision and Digital Image Processing
E-ISSN: 2986-8939 P-ISSN: 2986-8920
Vol. 4, No. 1, March 2026
DOI. https://doi.org/10.61255/decoding.v4i1.1693

A Statistical and Deployment-Aware Evaluation of EfficientNet-Lite0 and MobileNetV3-Small for 114-Class Lontara Script Recognition Performance

Ahmad Khairul Shiddiq✉
(Universitas Negeri Makassar, Makassar, Indonesia)

Ummul Khaeri Masna
(Universitas Negeri Makassar, Makassar, Indonesia)

Dwi Ihza Sari Akis
(Universitas Negeri Makassar, Makassar, Indonesia)

Sultan Budiluhur
(Universitas Negeri Makassar, Makassar, Indonesia)

OPEN ACCESS

ARTICLE HISTORY
Received: January 21, 2026
Revised: February 20, 2026
Accepted: March 17, 2026

KEYWORDS
EfficientNet-Lite0; Lontara recognition; MobileNetV3-Small; Model calibration; TensorFlow Lite.

ABSTRACT

Purpose: This study evaluates the accuracy, calibration, efficiency, and deployment trade-offs of EfficientNet-Lite0 and MobileNetV3-Small for 114-class Lontara script recognition.

Methods: A public Kaggle dataset containing 11,483 images was divided into training, validation, and test subsets using a stratified 70:15:15 split. Both transfer-learning pipelines were assessed using accuracy, macro and weighted metrics, confusion matrices, Expected Calibration Error, confidence-based rejection, McNemar's exact test, paired bootstrap confidence intervals, TensorFlow Lite conversion, post-training quantization, and preliminary CPU inference diagnostics.

Findings: EfficientNet-Lite0 achieved 94.31% test accuracy and a macro F1-score of 0.9415, compared with 90.60% and 0.9023 for MobileNetV3-Small. The paired difference in correctness was statistically significant ($p=1.5893 \times 10^{-6}$), and the bootstrapped accuracy difference was 3.71 percentage points with a 95% confidence interval of 2.21 to 5.22. EfficientNet-Lite0 showed a lower observed calibration error, although the between-model ECE difference was not conclusive. MobileNetV3-Small produced substantially smaller and faster float16 TensorFlow Lite models, while several visually similar classes remained vulnerable to systematic high-confidence errors.

Research implications: Model selection should reflect whether recognition quality or computational efficiency is prioritized. Results are limited to one dataset split, one reported training run, and preliminary CPU diagnostics.

Originality: The study integrates statistical comparison, calibration, selective prediction, quantization, deployment diagnostics, and class-level error analysis within one Lontara recognition experiment.

Correspondence Author: ✉ shiddiq071203@gmail.com
To cite this article: Shiddiq, A. K., Masna, U. K., Akis, D. I. S., & Budiluhur, S. (2026). A Statistical and Deployment-Aware Evaluation of EfficientNet-Lite0 and MobileNetV3-Small for 114-Class Lontara Script Recognition Performance. *Journal of Deep Learning, Computer Vision and Digital Image Processing, 4*(1), 82–102. https://doi.org/10.61255/decoding.v4i1.1693

This is an open access article under the CC BY-SA license

INTRODUCTION

Lontara is an indigenous writing system historically associated with the Bugis and Makassar communities of South Sulawesi, Indonesia. Its preservation is not limited to maintaining written symbols as cultural artifacts, but also requires technological mechanisms that allow the script to be recognized, processed, searched, and integrated into contemporary digital applications [1]–[3]. Automatic recognition can support the transcription of manuscripts, the development of educational applications, and broader access to written cultural resources [4], [5]. Earlier computational studies have therefore approached Lontara recognition through optical character recognition, feature extraction, image processing, and machine-learning classification.

The recognition of Lontara characters presents a nontrivial computer vision problem. Character identity may depend on localized visual elements, including small strokes, orientations, and diacritic configurations. Image quality, writing variability, segmentation quality, and similarity between character structures can consequently affect recognition performance. Ferandji et al. [6], for example, demonstrated that image quality influenced the performance of edge-detection operators applied to Lontara manuscript images. Areni et al. [7] combined Modified Direction Feature and Fourier Descriptor extraction with a support vector machine classifier for 23 Lontara characters, reporting 96% recognition accuracy. These studies showed that the representation of local shape and directional information is important for distinguishing Lontara characters.

The transition from handcrafted features to deep learning extended the capacity of Lontara recognition systems to learn hierarchical representations directly from image data. Hidayat et al. [8] combined contour-based segmentation, a sliding-window procedure, and a convolutional neural network to recognize 139 character classes formed from basic characters, diacritic combinations, and one additional character. The system achieved 96% accuracy on the evaluated dataset. This study established that CNN-based feature learning could support relatively large Lontara class inventories, but its primary evaluation remained centered on aggregate recognition accuracy and the overall performance of the segmentation-classification pipeline.

Research on other handwritten and historical scripts confirms that multiclass character recognition becomes increasingly difficult when datasets contain many visually similar classes, limited samples, heterogeneous writing styles, or imbalanced class distributions. Jalali et al. [9] examined ancient Korean-Chinese cursive character recognition across 4,795 classes and identified limited samples, handwriting variability, damaged characters, class imbalance, and interclass similarity as central recognition problems. Their results demonstrated the importance of transfer learning, attention mechanisms, and class-sensitive evaluation for scripts with a large and visually complex character inventory. Similarly, Al-Al-Haija [10] showed that transfer learning with a lightweight ShuffleNet backbone could provide high recognition performance while maintaining relatively low inference cost in multiclass handwritten character recognition. These findings support the use of pretrained lightweight CNNs when recognition quality and computational feasibility must be considered simultaneously [9], [10].

However, the practical value of a character-recognition model cannot be determined from accuracy alone. A model intended for mobile or educational use must operate under restrictions involving storage, memory, computation, and response time [11]–[13]. Increasing network capacity or input resolution may improve recognition, but it may also increase model size and inference latency. Conversely, a smaller network may be easier to deploy while sacrificing class-level discrimination.

MobileNetV3-Small and EfficientNet-Lite0 represent two relevant alternatives for examining this trade-off. MobileNetV3 was developed through hardware-aware neural architecture search and NetAdapt, with the Small configuration directed toward low-resource use cases and mobile CPU execution [14], [15]. The EfficientNet family applies compound scaling to balance network depth, width, and input resolution instead of increasing these dimensions independently [16]. EfficientNet-Lite variants subsequently adapted this accuracy-efficiency principle for resource-constrained inference. The two model families therefore embody different design priorities. MobileNetV3-Small places strong emphasis on hardware-aware efficiency, whereas EfficientNet-Lite0 seeks a balanced

scaling configuration that can retain discriminative performance with moderate computational requirements [14], [16]. Both publications are represented as Scopus-indexed records in subsequent indexed literature.

Even when a recognition model produces high classification accuracy, its confidence values may not be reliable. Confidence reliability concerns whether a model's predicted probability is consistent with its empirical likelihood of being correct. A model can achieve good discrimination while remaining overconfident or underconfident on particular observations. Sambyal et al. [17] showed that calibration constitutes a distinct aspect of deep-neural-network evaluation and that architecture, training regime, and representation learning can affect the relationship between confidence and correctness. Meir et al. (2024) similarly demonstrated that maximum output confidence does not fully represent the information available for distinguishing reliable from unreliable predictions. Calibration should therefore be evaluated separately from classification accuracy, particularly when confidence values are intended to guide operational decisions (Meir et al., 2024; [17], [18].

Expected Calibration Error and reliability diagrams provide practical mechanisms for examining the correspondence between confidence and observed correctness. Nevertheless, calibration estimates must be interpreted carefully because they are affected by sample size, binning decisions, class distribution, and the selected evaluation metric. Guilbert et al. [19] and Huang et al. [20] demonstrated that calibration methods and calibration measures may behave differently under imbalanced classification conditions. Although the present Lontara dataset is relatively balanced at the class level, this evidence remains relevant because confidence distributions and error frequencies may differ substantially across character classes. Reporting a single aggregate confidence value without examining calibration and class-specific error patterns would therefore provide an incomplete account of model reliability.

Confidence values can also be used to implement selective classification. Under this approach, a model automatically accepts predictions that exceed a confidence threshold while rejecting uncertain predictions for further processing or human review. The resulting performance must be evaluated through the relationship between coverage and accepted accuracy. Higher thresholds may increase the accuracy of retained predictions, but they also reduce the proportion of inputs processed automatically. Carse et al. [21] demonstrated that selective classification should be evaluated according to the consequences of rejection and the costs associated with different errors, rather than being interpreted as an unconditional improvement in model performance. In Lontara recognition, this distinction is relevant because the acceptable balance between coverage and error may differ between educational assistance, automatic transcription, and archival applications.

Deployment-oriented evaluation also requires examining how model conversion and quantization affect model behavior. Post-training quantization reduces the numerical precision used to represent model parameters, potentially decreasing storage and computational requirements without repeating the complete training process. However, reduced file size does not automatically guarantee lower latency or preserved recognition accuracy. Lee et al. [22] showed that the effects of post-training quantization vary across CNN architectures and hardware configurations, including relatively fragile lightweight models such as MobileNet, SqueezeNet, and ShuffleNet. Their hardware-based experiments also demonstrated that quantization configurations must be evaluated empirically because accuracy loss and execution efficiency depend on the interaction between the model, quantization scheme, compiler, and target device.

A valid comparison between two recognition pipelines must additionally consider the dependency between their predictions. When both classifiers are tested on the same images, their outcomes are paired rather than independent. Comparing two accuracy percentages alone does not determine whether one model systematically classifies more instances correctly. McNemar's test is appropriate when two trained classifiers are evaluated once on the same test observations because the test examines discordant prediction pairs [23], [24]. This procedure distinguishes cases correctly classified only by the first model from those correctly classified only by the second model. Statistical comparison is particularly relevant when an observed accuracy difference is used to support a claim that one recognition pipeline performs more effectively than another on a specific test set.

The Lontara studies identified in the present review have contributed important evidence regarding handcrafted feature extraction, edge-based preprocessing, segmentation, and CNN classification [6]–[8]. However, these studies have not jointly evaluated lightweight CNN pipelines through classification performance, paired statistical comparison, confidence calibration, selective-prediction behavior, post-training quantization, model size, inference diagnostics, and class-specific error structures. The unresolved problem is therefore not merely whether another CNN can recognize Lontara characters. The more consequential question is whether a recognition pipeline that performs well in aggregate also provides statistically defensible, confidence-aware, class-sensitive, and computationally feasible behavior for resource-constrained use.

To address this problem, the present study compares EfficientNet-Lite0 and MobileNetV3-Small in a 114-class Lontara character-recognition task. The two models are evaluated using the same training, validation, and test partitions. The evaluation includes aggregate and macro-averaged classification metrics, paired instance-level statistical comparison, confidence calibration, confidence-threshold analysis, TensorFlow Lite conversion, post-training quantization, preliminary CPU inference diagnostics, and class-level misclassification analysis. Because the implemented pipelines differ in input resolution, preprocessing, backbone implementation, and fine-tuning configuration, the study is positioned as a comparison of complete transfer-learning pipelines rather than a strictly controlled test of architectural superiority. This qualification is necessary because any observed performance difference may reflect the combined effects of architecture, image resolution, preprocessing, and optimization configuration.

The study provides three principal contributions. First, it presents an integrated evaluation of two lightweight transfer-learning pipelines in a relatively large 114-class Lontara recognition problem. Second, it supplements descriptive performance measures with paired statistical evidence, confidence calibration, threshold-specific coverage, and bootstrap uncertainty estimates. Third, it connects recognition quality with deployment-related characteristics by examining TensorFlow Lite model size, quantization behavior, preliminary inference time, and systematic high-confidence errors. The contribution lies in the scope and rigor of the evaluation framework, not in proposing a new CNN architecture.

Accordingly, the study addresses the following research questions:

RQ1: How accurately do the implemented EfficientNet-Lite0 and MobileNetV3-Small pipelines classify Lontara character images across 114 classes?

RQ2: Is there a statistically significant difference in instance-level correctness between the two pipelines when they are evaluated on the same test images?

RQ3: What trade-offs emerge among classification performance, confidence calibration, selective-prediction coverage, model size, quantization format, and preliminary CPU inference efficiency?

RQ4: Which Lontara character classes and directional misclassification pairs remain most vulnerable, and what error patterns can be identified from the predictions?

METHOD

Research Design

This study used an experimental computer vision design to compare MobileNetV3-Small and EfficientNet-Lite0 transfer-learning pipelines for 114-class Lontara script recognition. Both pipelines used identical dataset partitions, evaluation procedures, confidence and calibration analyses, TensorFlow Lite conversion, and paired statistical validation. Because the pipelines differed in input resolution, preprocessing, backbone accessibility, and fine-tuning strategy, the experiment was treated as a deployment-aware pipeline comparison rather than a strictly architecture-isolated benchmark. Figure 1 summarizes the experimental workflow.

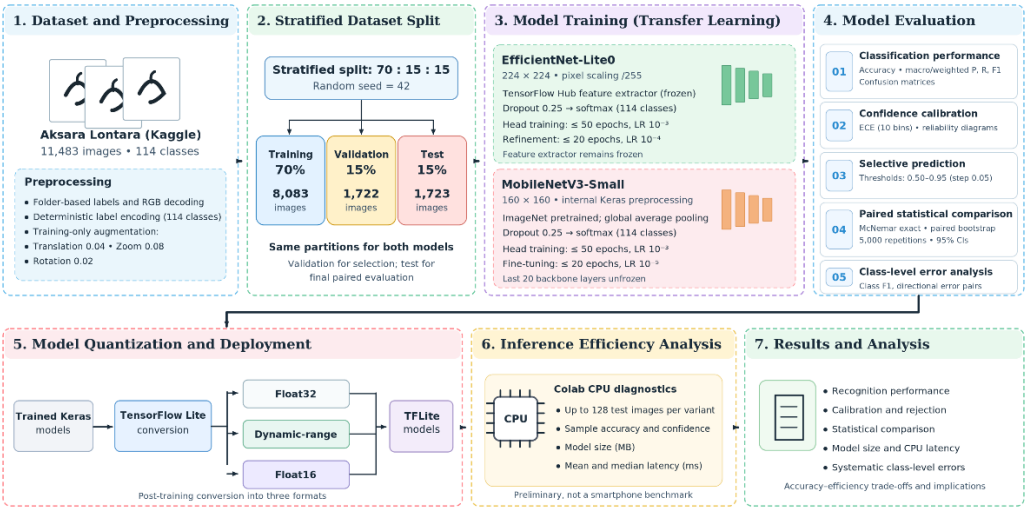


Figure 1. Experimental workflow

Dataset Preparation and Preprocessing

The image data were obtained from the publicly available *Aksara Lontara* dataset uploaded by Arsan5623 on Kaggle (Arsan5623, n.d.). The aksara-lontara-fix directory was used as the source directory for the experiment. It contained 11,483 Lontara character images distributed across 114 class-specific folders. Each class contained between 100 and 113 images, with an average of approximately 100.7 images per class.

The dataset was organized into class folders and included JPG, JPEG, PNG, BMP, and WEBP images. Labels were obtained from parent-folder names, alphabetically encoded, and recorded in a deterministic manifest sorted by label and filename, producing 114 classes. Stratified random sampling divided the dataset into training, validation, and test subsets at a 70:15:15 ratio. A fixed seed of 42 was applied to Python, NumPy, and TensorFlow.

Images were decoded as three-channel RGB tensors, converted to floating-point format, batched at 8, and prefetched with a buffer size of 1 using two parallel loading calls. MobileNetV3-Small images were resized to 160 × 160 pixels and processed internally by the Keras backbone. EfficientNet-Lite0 images were resized to 224 × 224 pixels and divided by 255 before entering the TensorFlow Hub layer. Training-only augmentation comprised random translation of 0.04, random zoom of 0.08, and random rotation of 0.02.

Model Development and Training

MobileNetV3-Small used tf.keras.applications.MobileNetV3Small with ImageNet weights, include_top=False, and include_preprocessing=True. MobileNetV3 was originally designed through hardware-aware neural architecture search and NetAdapt for mobile execution [14]. The backbone was followed by global average pooling, dropout of 0.25, and a 114-unit softmax classifier. The frozen-backbone classification head was trained for up to 50 epochs using Adam, a learning rate of 1×10^{-3}, and sparse categorical cross-entropy. The best checkpoint was subsequently reloaded, and only the final 20 backbone layers were unfrozen and fine-tuned for up to 20 epochs at 1×10^{-5}.

EfficientNet-Lite0 used the TensorFlow Hub model efficientnet/lite0/feature-vector/2, followed by dropout of 0.25 and a 114-unit softmax classifier. The EfficientNet family was developed by jointly balancing network depth, width, and input resolution [16]. The frozen feature extractor and classifier head were trained for up to 50 epochs using Adam at 1×10^{-3}. The best checkpoint was refined for up to 20 epochs at 1×10^{-4}; because FINE_TUNE_FEATURE_EXTRACTOR=False, the feature extractor remained frozen. Both pipelines used validation-accuracy checkpointing, validation-loss learning-rate reduction, early stopping, and CSV training logs.

For N samples, sparse categorical cross-entropy was calculated as:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} \log(p_{i,y_i}) \qquad (1)$$

where N denotes the number of training samples, and p_{i,y_i} denotes the predicted probability assigned to the true class of sample i.

Model Evaluation

Validation and test predictions retained the true label, predicted label, and maximum softmax confidence. Performance was evaluated using accuracy, macro precision, macro recall, macro F1-score, weighted precision, weighted recall, and weighted F1-score. Class-level reports and confusion matrices were used to identify vulnerable classes and frequent misclassification pairs.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad (2)$$

$$Precision = \frac{TP}{TP + FP} \qquad (3)$$

$$Recall = \frac{TP}{TP + FN} \qquad (4)$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \qquad (5)$$

For C classes, macro and weighted F1-scores were calculated as:

$$Macro\ F1 = \frac{1}{C}\sum_{c=1}^{C} F1_c \qquad (6)$$

$$Weighted\ F1 = \sum_{c=1}^{C} \frac{n_c}{N} F1_c \qquad (7)$$

where C denotes the number of classes, $F1_c$ denotes the F1-score of class c, n_c denotes the number of samples in class c, and N denotes the total number of samples.

Confidence, Calibration, and Deployment Diagnostics

Maximum softmax probability was used as the confidence score. Thresholds from 0.50 to 0.95 at increments of 0.05 were evaluated. For threshold ttt, the accepted set was:

$$A_t = \{i \mid conf_i \geq t\} \qquad (8)$$

Coverage, rejection rate, and accepted accuracy were calculated as:

$$Coverage(t) = \frac{|A_t|}{N} \qquad (9)$$

$$\boldsymbol{Rejection}(\boldsymbol{t}) = \mathbf{1} - \boldsymbol{Coverage}(\boldsymbol{t}) \qquad (10)$$

$$\boldsymbol{Accepted\ Accuracy}(\boldsymbol{t}) = \frac{\mathbf{1}}{\mid \boldsymbol{A_t} \mid} \sum_{\boldsymbol{i} \in \boldsymbol{A_t}} \mathbb{1}(\hat{\boldsymbol{y}}_{\boldsymbol{i}} = \boldsymbol{y}_{\boldsymbol{i}}) \qquad (11)$$

where $\mathbb{1}(\cdot)$is an indicator function.

Calibration was evaluated using reliability diagrams and Expected Calibration Error with 10 equal-width confidence bins. ECE measures the weighted absolute difference between empirical accuracy and mean confidence [25]:

$$\boldsymbol{ECE} = \sum_{\boldsymbol{m}=\mathbf{1}}^{\boldsymbol{M}} \frac{\mid \boldsymbol{B_m} \mid}{\boldsymbol{N}} \mid \boldsymbol{acc}(\boldsymbol{B_m}) - \boldsymbol{conf}(\boldsymbol{B_m}) \mid \qquad (12)$$

where $\boldsymbol{M}$ is the number of bins, and $\boldsymbol{B_m}$ is confidence bin $\boldsymbol{m}$,

Each Keras model was converted into float32, dynamic-range quantized, and float16 TFLite variants. Post-training quantization is intended to reduce model size and computational requirements, although its effects must be empirically evaluated for each model [26]. Resource-variable support was enabled for EfficientNet-Lite0. Up to 128 test images were used to record sample accuracy, mean confidence, mean and median Colab-CPU inference time, and input and output types. The inference measurements were treated as preliminary diagnostics rather than direct smartphone benchmarks.

$$\boldsymbol{Size\ Reduction}(\%) = \frac{\boldsymbol{Size_{Keras}} - \boldsymbol{Size_{TFLite}}}{\boldsymbol{Size_{Keras}}} \times \mathbf{100} \qquad (13)$$

where $\boldsymbol{Size_{Keras}}$denotes the size of the saved Keras model and $\boldsymbol{Size_{TFLite}}$denotes the size of the converted TensorFlow Lite model.

Paired Statistical Validation

Predictions were aligned using true labels and filenames to ensure that both models were evaluated on identical paired samples. McNemar's exact test compared instance-level correctness because paired classifier outputs should be evaluated through their discordant outcomes rather than as independent proportions [23]:

	MobileNet correct	MobileNet wrong	
EfficientNet correct	$\boldsymbol{a}$	$\boldsymbol{b}$	(14)
EfficientNet wrong	$\boldsymbol{c}$	$\boldsymbol{d}$	

Here, $\boldsymbol{a}$ denotes samples correctly classified by both models, $\boldsymbol{b}$ samples correctly classified only by EfficientNet-Lite0, $\boldsymbol{c}$ samples correctly classified only by MobileNetV3-Small, and $\boldsymbol{d}$ samples misclassified by both.

The exact test was applied to $\boldsymbol{b}$ and $\boldsymbol{c}$ The continuity-corrected statistic and Haldane-corrected matched odds ratio were expressed as:

$$\boldsymbol{\chi^2} = \frac{(\mid \boldsymbol{b} - \boldsymbol{c} \mid - \mathbf{1})^2}{\boldsymbol{b} + \boldsymbol{c}} \qquad (15)$$

$$\boldsymbol{OR_{matched}} = \frac{\boldsymbol{b} + \mathbf{0.5}}{\boldsymbol{c} + \mathbf{0.5}} \qquad (16)$$

where 0.5 was added to prevent an undefined ratio when one discordant cell was zero.

Paired bootstrap confidence intervals used 5,000 repetitions with seed 42 and resampling over paired sample indices. Percentile-based 95% confidence intervals were estimated for differences in accuracy, macro F1, weighted F1, ECE, and threshold-specific coverage and accepted accuracy at thresholds of 0.70, 0.80, and 0.90.

$$\boldsymbol{\Delta_r = S_{Eff,r} - S_{Mob,r}} \qquad (17)$$

$$\boldsymbol{CI_{95\%} = [Q_{0.025}(\Delta), Q_{0.975}(\Delta)]} \qquad (18)$$

where $\boldsymbol{S_{Eff,r}}$ and $\boldsymbol{S_{Mob,r}}$ are the evaluated statistics in bootstrap repetition $\boldsymbol{r}$ while $\boldsymbol{Q_{0.025}}$ and $\boldsymbol{Q_{0.975}}$ **are** the 2.5th and 97.5th percentiles.

RESULTS AND DISCUSSION

Results

Dataset Distribution and Experimental Output

The final dataset comprised 11,483 Lontara character images distributed across 114 classes. Following stratified partitioning at a ratio of 70:15:15, 8,083 images were assigned to the training subset, 1,722 images to the validation subset, and 1,723 images to the test subset. Each class contained between 100 and 113 images, with an average of approximately 100.7 images per class.

EfficientNet-Lite0 and MobileNetV3-Small were trained and evaluated using identical class labels and dataset partitions. Their test predictions were aligned using the true class label and filename, resulting in 1,723 paired test observations. No unmatched predictions or inconsistently aligned labels were identified.

Table 1. Distribution of the Lontara character dataset

Dataset subset	Number of images	Percentage	Number of classes
Training	8083	70%	114
Validation	1722	15%	114
Test	1723	15%	114
Total	11,483	100%	114

Training and Validation Performance

The training and validation histories of both pipelines are presented in Figure 2. During classifier-head training, MobileNetV3-Small reached its highest validation accuracy of 90.1% at epoch 50, with a corresponding validation loss of 0.34. After the final 20 backbone layers were partially fine-tuned, its best validation accuracy increased to 90.6% at epoch 54.

EfficientNet-Lite0 reached a maximum validation accuracy of 94.0% at epoch 18 during classifier-head training. Because the TensorFlow Hub feature extractor remained frozen during the refinement stage, only the classification head was further optimized. The final EfficientNet-Lite0 checkpoint achieved a validation accuracy of 93.8% and a validation loss of 0.19.

The final training-to-validation accuracy difference was 4.4 percentage points for MobileNetV3-Small and 4.9 percentage points for EfficientNet-Lite0.

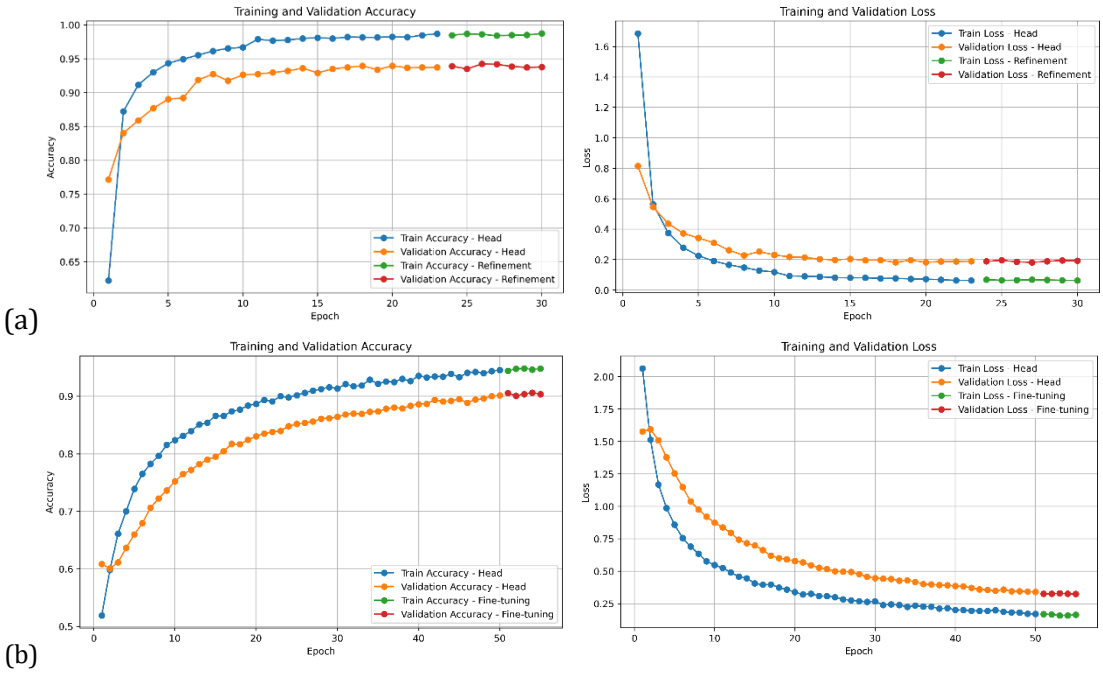


Figure 2. Training and validation accuracy and loss curves for (a) EfficientNet-Lite0 and (b) MobileNetV3-Small

Overall Classification Performance

Table 2 presents the validation and test performance of both recognition pipelines. On the test subset, EfficientNet-Lite0 achieved an accuracy of 0.9431, compared with 0.9060 for MobileNetV3-Small. The absolute accuracy difference was therefore 0.0371, or 3.71 percentage points.

EfficientNet-Lite0 produced a macro F1-score of 0.9415 and a weighted F1-score of 0.9413. MobileNetV3-Small produced corresponding values of 0.9023 and 0.9025. EfficientNet-Lite0 also recorded higher macro precision, macro recall, weighted precision, and weighted recall on both the validation and test subsets.

Table 2. Validation and test classification performance

Dataset	Model	Accuracy	Macro precision	Macro recall	Macro F1	Weighted precision	Weighted recall	Weighted F1
Validation	EfficientNet-Lite0	0.9425	0.9511	0.9428	0.9410	0.9514	0.9425	0.9409
Validation	MobileNetV3-Small	0.9059	0.9202	0.9056	0.9040	0.9204	0.9059	0.9044
Test	EfficientNet-Lite0	0.9431	0.9558	0.9436	0.9415	0.9560	0.9431	0.9413
Test	MobileNetV3-Small	0.9060	0.9160	0.9058	0.9023	0.9161	0.9060	0.9025

Paired Statistical Comparison

The paired prediction outcomes are presented in Table 3. Of the 1,723 test images, 1,505 were correctly classified by both pipelines, while 42 were incorrectly classified by both. EfficientNet-Lite0 alone correctly classified 120 images that MobileNetV3-Small misclassified. Conversely, MobileNetV3-Small alone correctly classified 56 images that EfficientNet-Lite0 misclassified.

Table 3. Paired prediction contingency table

	MobileNetV3-Small correct	MobileNetV3-Small incorrect
EfficientNet-Lite0 correct	1,505	120
EfficientNet-Lite0 incorrect	56	42

McNemar's exact test produced a p-value of of 1.5893×10^{-6}. At α=0.05\alpha=0.05α=0.05, the difference in instance-level correctness was statistically significant. The matched odds ratio with Haldane correction was 2.13, with an approximate 95% confidence interval of [1.55, 2.93].

The paired bootstrap analysis estimated an accuracy difference of 0.0371, with a 95% confidence interval of [0.0221, 0.0522]. The macro F1 difference was 0.0392, with a confidence interval of [0.0249, 0.0559], while the weighted F1 difference was 0.0389, with a confidence interval of [0.0233, 0.0545]. All three intervals excluded zero.

The observed ECE difference was −0.0083, with a 95% confidence interval of [−0.0197, 0.0036]. Because this interval included zero, the difference in ECE was not statistically conclusive.

Table 4. Paired statistical comparison between EfficientNet-Lite0 and MobileNetV3-Small

Statistical measure	Estimate	95% confidence interval	p-value
Accuracy difference	0.0371	[0.0221, 0.0522]	—
Macro F1 difference	0.0392	[0.0249, 0.0559]	—
Weighted F1 difference	0.0389	[0.0233, 0.0545]	—
ECE difference	−0.0083	[−0.0197, 0.0036]	—
McNemar's exact test	—	—	1.5893×10^{-6}
Matched odds ratio with Haldane correction	2.13	[1.55, 2.93]	—

Positive differences indicate higher values for EfficientNet-Lite0. For ECE, a negative difference indicates that EfficientNet-Lite0 had a lower observed calibration error because smaller ECE values represent closer agreement between confidence and empirical accuracy.

Confidence Distribution and Calibration

The confidence-score distributions are presented in Figure 3. EfficientNet-Lite0 produced an overall mean maximum softmax confidence of 0.9381, while MobileNetV3-Small produced a mean confidence of 0.9152.

For correctly classified observations, the mean confidence values were 0.9521 for EfficientNet-Lite0 and 0.9413 for MobileNetV3-Small. For incorrectly classified observations, the corresponding mean confidence values were 0.7057 and 0.6639.

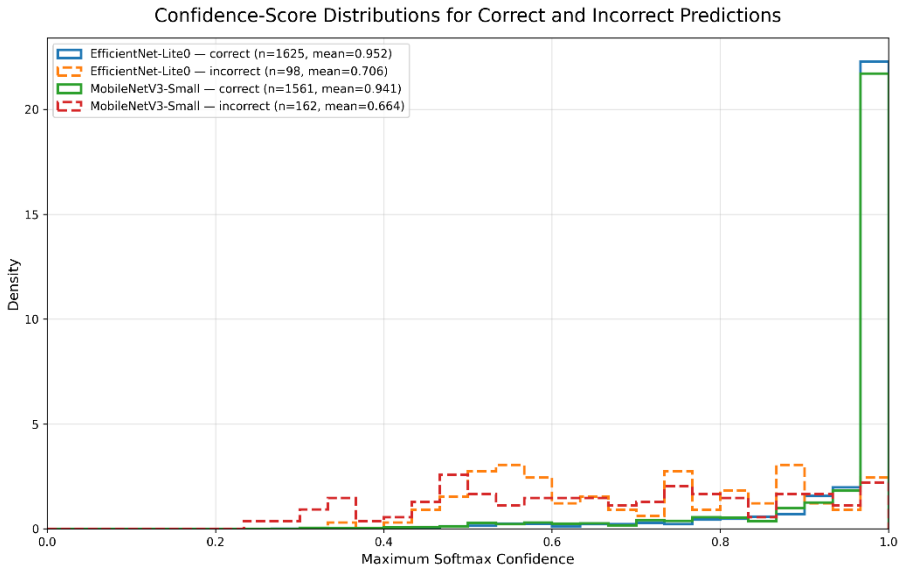


Figure 3. Confidence-score distributions for correct and incorrect predictions

EfficientNet-Lite0 obtained an ECE of 0.0087, whereas MobileNetV3-Small obtained an ECE of 0.0170. The absolute observed difference was 0.0083 in favor of EfficientNet-Lite0, although the paired bootstrap confidence interval reported in Table 4 included zero.

Figure 4 presents reliability diagrams based on 10 equal-width confidence bins. For EfficientNet-Lite0, the largest absolute calibration gap was 0.713 in the 0.20 to 0.30 confidence bin. MobileNetV3-Small produced a maximum calibration gap of 0.272 in the same confidence interval.

Because the number of observations within each confidence bin was not reported, these bin-specific calibration gaps should be interpreted descriptively.

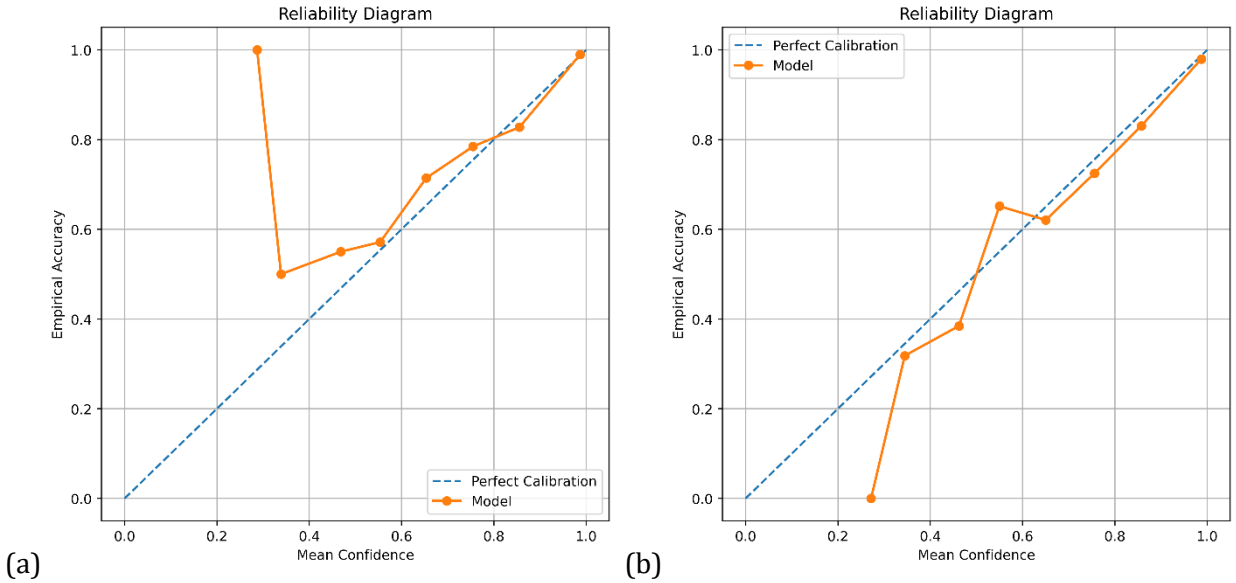


Figure 4. Reliability diagrams for (a) EfficientNet-Lite0 and (b) MobileNetV3-Small

Confidence-Threshold Performance

Table 5 summarizes the effects of confidence-based rejection. For both pipelines, higher confidence thresholds reduced coverage and increased the accuracy of accepted predictions.

At a threshold of 0.50, EfficientNet-Lite0 accepted 1,700 predictions, corresponding to a coverage of 98.67% and an accepted accuracy of 94.82%. MobileNetV3-Small accepted 1,658 predictions, corresponding to a coverage of 96.23% and an accepted accuracy of 92.82%.

At a threshold of 0.80, EfficientNet-Lite0 retained 1,530 observations, producing a coverage of 88.80% and an accepted accuracy of 97.71%. MobileNetV3-Small retained 1,436 observations, producing a coverage of 83.34% and an accepted accuracy of 96.73%.

At the highest evaluated threshold of 0.95, EfficientNet-Lite0 retained 74.81% of the test set and achieved an accepted accuracy of 99.15%. MobileNetV3-Small retained 70.17% of the test set and achieved an accepted accuracy of 98.68%.

Table 5. Confidence-threshold performance on the test subset

Threshold	Model	Accepted samples	Rejected samples	Coverage	Rejection rate	Accepted accuracy
0.50	EfficientNet-Lite0	1,700	23	98.67%	1.33%	94.82%
0.50	MobileNetV3-Small	1,658	65	96.23%	3.77%	92.82%
0.60	EfficientNet-Lite0	1,637	86	95.01%	4.99%	96.27%
0.60	MobileNetV3-Small	1,592	131	92.40%	7.60%	93.97%
0.70	EfficientNet-Lite0	1,595	128	92.57%	7.43%	96.93%
0.70	MobileNetV3-Small	1,534	189	89.03%	10.97%	95.18%

0.80	EfficientNet-Lite0	1,530	193	88.80%	11.20%	97.71%
0.80	MobileNetV3-Small	1,436	287	83.34%	16.66%	96.73%
0.90	EfficientNet-Lite0	1,414	309	82.07%	17.93%	98.94%
0.90	MobileNetV3-Small	1,318	405	76.49%	23.51%	97.95%
0.95	EfficientNet-Lite0	1,289	434	74.81%	25.19%	99.15%
0.95	MobileNetV3-Small	1,209	514	70.17%	29.83%	98.68%

The paired bootstrap results for thresholds of 0.70, 0.80, and 0.90 are presented in Table 6. At a threshold of 0.70, EfficientNet-Lite0 produced a coverage advantage of 0.0354 and an accepted-accuracy advantage of 0.0175. Both confidence intervals excluded zero.

At a threshold of 0.80, the coverage difference was 0.0546, with a confidence interval of [0.0331, 0.0749]. The accepted-accuracy difference was 0.0099, with a confidence interval of [−0.0016, 0.0218]. The coverage difference was conclusive, while the accepted-accuracy difference was not.

At a threshold of 0.90, the coverage difference was 0.0557 and the accepted-accuracy difference was 0.0099. Both confidence intervals excluded zero.

Table 6. Bootstrap confidence intervals for confidence-threshold differences

Threshold	Measure	Difference: EfficientNet – MobileNet	95% confidence interval
0.70	Coverage	0.0354	[0.0180, 0.0534]
0.70	Accepted accuracy	0.0175	[0.0048, 0.0299]
0.80	Coverage	0.0546	[0.0331, 0.0749]
0.80	Accepted accuracy	0.0099	[−0.0016, 0.0218]
0.90	Coverage	0.0557	[0.0319, 0.0795]
0.90	Accepted accuracy	0.0099	[0.0008, 0.0193]

Note: Differences are reported on a 0–1 scale and were estimated using 5,000 paired bootstrap repetitions. Positive values indicate higher performance for EfficientNet-Lite0.

TensorFlow Lite Conversion and Model Size

Both trained models were successfully converted into float32, dynamic-range quantized, and float16 TensorFlow Lite variants.

The original EfficientNet-Lite0 Keras model occupied 14.79 MB. Its float32, dynamic-range, and float16 TensorFlow Lite variants occupied 13.36 MB, 3.67 MB, and 6.71 MB, respectively. These values corresponded to size reductions of 9.67%, 75.20%, and 54.67%.

The original MobileNetV3-Small Keras model occupied 7.56 MB. Its float32, dynamic-range, and float16 TensorFlow Lite variants occupied 3.81 MB, 1.12 MB, and 1.94 MB, respectively. The corresponding size reductions were 49.63%, 85.17%, and 74.35%.

The smallest converted model was the dynamic-range MobileNetV3-Small variant at 1.12 MB. The largest converted model was the EfficientNet-Lite0 float32 variant at 13.36 MB.

Table 7. Model size and TensorFlow Lite conversion results

Model	Format	File size (MB)	Size reduction (%)	Input type	Output type
EfficientNet-Lite0	Keras	14.79	NA	float32	float32
EfficientNet-Lite0	TFLite float32	13.36	9.67%	float32	float32
EfficientNet-Lite0	TFLite dynamic range	3.67	75.20%	float32	float32
EfficientNet-Lite0	TFLite float16	6.71	54.67%	float32	float32
MobileNetV3-Small	Keras	7.56	NA	float32	float32

MobileNetV3-Small	TFLite float32	3.81	49.63%	float32	float32
MobileNetV3-Small	TFLite dynamic range	1.12	85.17%	float32	float32
MobileNetV3-Small	TFLite float16	1.94	74.35%	float32	float32

TensorFlow Lite Inference Diagnostics

The preliminary TensorFlow Lite diagnostic was conducted using 128 test images for each converted variant.

For EfficientNet-Lite0, the float16 variant produced the lowest mean inference time at 12.130 ms per image and the lowest median inference time at 11.537 ms. Its sample accuracy was 93.75%, identical to the float32 and dynamic-range variants.

For MobileNetV3-Small, the float16 variant produced the lowest mean inference time at 1.446 ms and the lowest median inference time at 1.282 ms. Its sample accuracy was 89.84%, identical to the float32 variant. The dynamic-range variant achieved a sample accuracy of 69.53%, which was 20.31 percentage points lower than its float32 counterpart.

The MobileNetV3-Small float16 variant recorded the shortest mean inference time among all evaluated variants. The EfficientNet-Lite0 dynamic-range variant recorded the longest mean inference time at 27.397 ms.

Table 8. TensorFlow Lite inference diagnostics on up to 128 test samples

Model	TFLite format	Sample accuracy	Mean confidence	Mean inference time (ms)	Median inference time (ms)
EfficientNet-Lite0	Float32	93.75%	0.9521	12.770	11.580
EfficientNet-Lite0	Dynamic range	93.75%	0.9527	27.397	26.562
EfficientNet-Lite0	Float16	93.75%	0.9522	12.130	11.537
MobileNetV3-Small	Float32	89.84%	0.9146	1.529	1.336
MobileNetV3-Small	Dynamic range	69.53%	0.8127	2.693	2.423
MobileNetV3-Small	Float16	89.84%	0.9157	1.446	1.282

These values represent preliminary measurements obtained in the experimental CPU environment and not direct smartphone benchmark results.

Class-Level Performance

The class-level results showed substantial variation across the 114 character classes. EfficientNet-Lite0 produced F1-scores ranging from 0.2353 to 1.0000, with a median of 1.0000. MobileNetV3-Small produced F1-scores ranging from 0.2000 to 1.0000, with a median of 0.9333.

EfficientNet-Lite0 achieved an F1-score of at least 0.95 in 82 classes, while MobileNetV3-Small reached this level in 46 classes. Ten classes produced an F1-score below 0.80 for EfficientNet-Lite0, compared with 11 classes for MobileNetV3-Small.

For EfficientNet-Lite0, the five lowest-performing classes were Lo, Le, Wo, Ho, and O. The lowest F1-score occurred for Lo at 0.2353. Only 2 of the 15 Lo observations were correctly identified, while 13 were classified as Le.

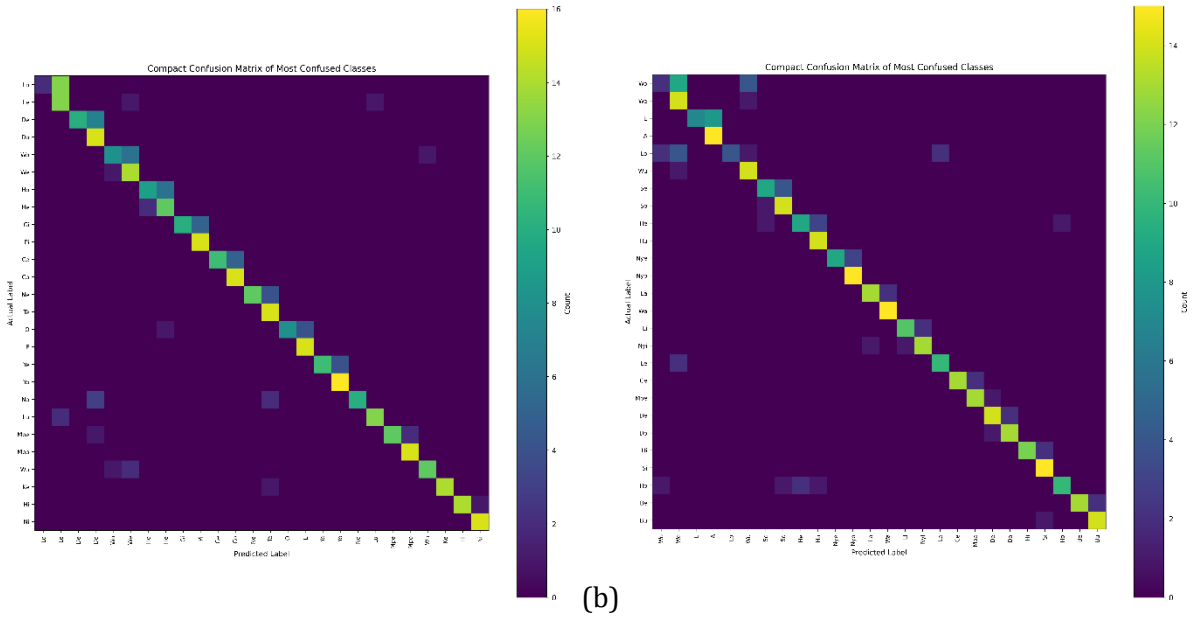
For MobileNetV3-Small, the five lowest-performing classes were Wo, Lo, E, We, and He. The Wo class obtained the lowest F1-score at 0.2000. Two of its 15 observations were correctly classified, while nine were classified as We and four as Wu.

Table 9. Lowest-performing character classes

Model	True class	Support	Precision	Recall	F1-score	Most frequent predicted class
EfficientNet-Lite0	Lo	15	1.0000	0.1333	0.2353	Le (13)
EfficientNet-Lite0	Le	15	0.4643	0.8667	0.6047	Lu and We (1 each)
EfficientNet-Lite0	Wo	15	0.8000	0.5333	0.6400	We (6)
EfficientNet-Lite0	Ho	15	0.8182	0.6000	0.6923	He (6)
EfficientNet-Lite0	O	15	1.0000	0.5333	0.6957	E (4)
MobileNetV3-Small	Wo	15	0.4000	0.1333	0.2000	We (9)
MobileNetV3-Small	Lo	15	1.0000	0.2667	0.4211	We (4)
MobileNetV3-Small	E	15	0.7778	0.4667	0.5833	A (8)
MobileNetV3-Small	We	15	0.4667	0.9333	0.6222	Wu (1)
MobileNetV3-Small	He	15	0.8182	0.6000	0.6923	Hu (3)

Misclassification Patterns

Figure 5 presents compact confusion matrices for the two pipelines. EfficientNet-Lite0 produced 98 incorrect test predictions, while MobileNetV3-Small produced 162. Forty-two test observations were incorrectly classified by both models.



(a) (b)

Figure 5. Confusion matrices for (a) EfficientNet-Lite0 and (b) MobileNetV3-Small

For EfficientNet-Lite0, the most frequent directional error was Lo predicted as Le, which occurred in 13 observations. This was followed by De predicted as Do in seven observations. Wo predicted as We and Ho predicted as He each occurred six times.

For MobileNetV3-Small, the most frequent directional error was Wo predicted as We, which occurred nine times. This was followed by E predicted as A in eight observations. Lo predicted as We, Wo predicted as Wu, and Se predicted as So each occurred four times.

EfficientNet-Lite0 produced 15 incorrect predictions with confidence scores of at least 0.90, while MobileNetV3-Small produced 27. The highest-confidence EfficientNet-Lite0 error involved a No observation predicted as Do with a confidence score of 0.9928. The highest-confidence MobileNetV3-Small error involved a Pi observation predicted as Gi with a confidence score of 0.9971.

Table 10. Most frequent misclassification pairs

Rank	Model	True class	Predicted class	Error count	Mean confidence
1	EfficientNet-Lite0	Lo	Le	13	0.7274
2	EfficientNet-Lite0	De	Do	7	0.7544
3	EfficientNet-Lite0	Wo	We	6	0.7856
4	EfficientNet-Lite0	Ho	He	6	0.7629
5	EfficientNet-Lite0	Gi	Pi	5	0.7902
1	MobileNetV3-Small	Wo	We	9	0.7706
2	MobileNetV3-Small	E	A	8	0.8070
3	MobileNetV3-Small	Lo	We	4	0.6920
4	MobileNetV3-Small	Wo	Wu	4	0.6642
5	MobileNetV3-Small	Se	So	4	0.6273

Discussion

Comparative Classification Performance

The results provide a direct answer to RQ1 by showing that the implemented EfficientNet-Lite0 pipeline produced stronger overall and class-balanced recognition performance than MobileNetV3-Small. EfficientNet-Lite0 achieved a test accuracy of 94.31%, compared with 90.60% for MobileNetV3-Small, representing an absolute difference of 3.71 percentage points. The same pattern was observed for macro F1-score, where EfficientNet-Lite0 reached 0.9415 and MobileNetV3-Small reached 0.9023. The near-equivalence between macro and weighted F1-scores for EfficientNet-Lite0 also indicates that its aggregate performance was not primarily produced by a limited group of well-recognized classes.

The performance difference is consistent with the design orientation of the two model families. EfficientNet was developed through compound scaling that jointly balances network depth, width, and input resolution, while MobileNetV3-Small was explicitly optimized for low-resource and mobile CPU environments through hardware-aware neural architecture search and NetAdapt [14], [16]. EfficientNet-Lite0 may therefore retain richer feature representations at a higher computational cost, whereas MobileNetV3-Small prioritizes compactness and execution efficiency.

Nevertheless, the present experiment does not isolate architecture as the sole cause of the observed difference. EfficientNet-Lite0 received 224 × 224-pixel images, while MobileNetV3-Small received 160 × 160-pixel images. Their preprocessing implementations and fine-tuning procedures also differed. MobileNetV3-Small underwent partial backbone fine-tuning, whereas the EfficientNet-Lite0 feature extractor remained frozen. The results therefore establish that the implemented EfficientNet-Lite0 pipeline performed better under the reported configuration, but they do not demonstrate that EfficientNet-Lite0 would outperform MobileNetV3-Small under every resolution, preprocessing, or optimization setting.

The achieved accuracy is broadly consistent with earlier evidence that machine-learning and deep-learning methods can recognize Lontara characters effectively. Areni et al. [7] reported 96% accuracy using a combination of Modified Direction Feature, Fourier Descriptor, and support vector machine classification for 23 Lontara characters. Hidayat et al. [8] also reported 96% accuracy for a segmentation and CNN recognition pipeline involving 139 classes. However, these percentages are not directly comparable with the present results because the studies differed in dataset construction, class definitions, preprocessing, segmentation, training procedures, and evaluation protocols. The more defensible contribution of the present study is therefore not the claim that its accuracy is higher or lower than a previously reported value. Its contribution lies in combining test-set classification metrics with paired statistical comparison, calibration analysis, selective-prediction behaviour, quantization outcomes, and class-specific errors within the same 114-class experiment.

The learning curves further showed training-to-validation accuracy differences of 4.9 percentage points for EfficientNet-Lite0 and 4.4 percentage points for MobileNetV3-Small. These gaps indicate some reduction in generalization performance relative to the training data, although the validation and test metrics remained broadly consistent. Because the Kaggle dataset does not provide writer identities in the metadata used in this experiment, it is not possible to determine whether the models generalize to writers whose samples were completely excluded from training. This limitation is important in handwritten character recognition, where writing style, image degradation, class similarity, and limited representation of individual classes can materially influence recognition performance [9].

Statistical Evidence for the Difference Between Pipelines

RQ2 concerned whether the difference between the two pipelines was statistically supported on the same test images. The paired contingency table showed that both models correctly classified 1,505 images and both misclassified 42 images. EfficientNet-Lite0 correctly classified 120 images that MobileNetV3-Small misclassified, whereas MobileNetV3-Small correctly classified 56 images that EfficientNet-Lite0 misclassified. EfficientNet-Lite0 therefore produced a net gain of 64 correct predictions on the common test set.

McNemar's exact test yielded a p-value of 1.5893×10−61.5893 \times 10^{-6}1.5893×10−6, while the matched odds ratio was 2.13 with a 95% confidence interval of [1.55, 2.93]. The paired bootstrap estimate for the accuracy difference was 0.0371, with a confidence interval of [0.0221, 0.0522]. Confidence intervals for macro F1 and weighted F1 differences also excluded zero. The convergence of McNemar's test and the paired bootstrap results indicates that the difference was not limited to a descriptive comparison of two accuracy percentages.

The use of a paired test was methodologically appropriate because both pipelines classified exactly the same test observations. Treating the two accuracy values as independent proportions would ignore the dependency between predictions. McNemar's test specifically evaluates the discordant cases in which the classifiers produce different correctness outcomes on matched observations [23].

Statistical significance must nevertheless be interpreted within the scope of the design. The test establishes a difference between the two trained pipeline instances on the current 1,723-image test set. It does not quantify variability caused by alternative random initializations, training runs, data partitions, writers, or independently collected datasets. Because only one principal split and one reported training run per pipeline were used, the appropriate conclusion is that EfficientNet-Lite0 significantly outperformed MobileNetV3-Small on the paired test observations in this experiment. A claim of general architectural superiority would exceed the evidence.

Confidence Calibration and Selective Prediction

The calibration analysis addressed the reliability of the confidence values produced by the models. EfficientNet-Lite0 obtained an ECE of 0.0087, compared with 0.0170 for MobileNetV3-Small. Although the point estimates indicate lower observed calibration error for EfficientNet-Lite0, the paired bootstrap confidence interval for the ECE difference ranged from −0.0197 to 0.0036 and included zero. The evidence therefore does not establish a statistically conclusive calibration advantage.

This distinction matters because classification accuracy and confidence calibration measure different properties. A neural network may discriminate effectively between classes while producing probabilities that do not correspond closely to empirical correctness. Architectural characteristics, training regimes, regularization, and learned representations can influence calibration independently of accuracy [17], [25]. The present findings should therefore be expressed as lower observed ECE for EfficientNet-Lite0, not as definitive proof that it is better calibrated.

The reliability diagrams showed sizeable gaps in some low-confidence bins. However, the number of observations within each bin was not reported. Large gaps in sparsely populated bins can be unstable and should not receive the same interpretive weight as gaps calculated from densely populated confidence intervals. The overall ECE values and bootstrap interval consequently provide a more defensible basis for comparing calibration than the largest single-bin discrepancy.

Confidence-based rejection produced the expected trade-off between coverage and accepted accuracy. Increasing the threshold reduced the proportion of automatically accepted observations but increased accuracy among the retained predictions. At a threshold of 0.80, EfficientNet-Lite0 retained 88.80% of the test set with an accepted accuracy of 97.71%, while MobileNetV3-Small retained 83.34% with an accepted accuracy of 96.73%.

The paired bootstrap results qualify this comparison. At the 0.80 threshold, EfficientNet-Lite0 had a conclusive coverage advantage of 5.46 percentage points, but the confidence interval for the 0.99-percentage-point accepted-accuracy difference included zero. EfficientNet-Lite0 therefore retained more observations at this threshold, but its accepted-accuracy advantage was not statistically conclusive.

Selective prediction should not be interpreted as an unconditional improvement in classification. Higher accepted accuracy is achieved partly by removing lower-confidence observations from automatic processing. The relevant outcome is therefore the risk-coverage relationship rather than accepted accuracy alone [27].

A threshold of 0.80 may be described as a potentially useful exploratory operating point, but it should not be labelled optimal. The threshold was examined using test-set results, and no explicit cost function was defined for balancing rejection against misclassification. The appropriate threshold would also depend on the application. An educational application might prioritize coverage, whereas archival transcription could prioritize accuracy and direct rejected predictions to human verification.

Moreover, confidence rejection did not eliminate systematic high-confidence errors. EfficientNet-Lite0 generated 15 incorrect predictions with confidence values of at least 0.90, while MobileNetV3-Small generated 27. Maximum softmax confidence should therefore not be treated as proof of correctness, especially for repeatedly confused character pairs.

Model Size, Quantization, and Inference Trade-offs

The deployment analysis addressed RQ3 by demonstrating that predictive performance and computational efficiency produced different model rankings. EfficientNet-Lite0 achieved higher classification performance, whereas MobileNetV3-Small generated substantially smaller TensorFlow Lite models and faster inference in the preliminary CPU diagnostic.

Following float16 conversion, EfficientNet-Lite0 occupied 6.71 MB, compared with 1.94 MB for MobileNetV3-Small. MobileNetV3-Small float16 was therefore approximately 3.5 times smaller. Its mean inference time was 1.446 ms, compared with 12.130 ms for EfficientNet-Lite0 float16 in the same Colab CPU environment. This result is consistent with MobileNetV3-Small's original design objective of efficient execution under constrained computational resources [14].

The quantization results also show that maximum compression does not necessarily produce the best operational model. Dynamic-range quantization reduced MobileNetV3-Small to 1.12 MB, but its sample accuracy declined from 89.84% to 69.53%. The dynamic-range EfficientNet-Lite0 model retained the observed sample accuracy but was slower than its float32 and float16 variants. These outcomes are consistent with evidence that post-training quantization effects vary across architectures, operators, compiler configurations, and hardware targets. Model size reduction does not guarantee preserved accuracy or reduced latency because quantization and dequantization operations can introduce additional computational overhead [22].

Within the reported environment, float16 was the most balanced conversion format for both pipelines. It reduced file size, retained the observed 128-image sample accuracy, and produced the lowest mean inference time within each architecture. EfficientNet-Lite0 float16 is therefore the stronger candidate when recognition quality is prioritized, whereas MobileNetV3-Small float16 is the stronger candidate when storage and latency are more restrictive.

These conclusions remain preliminary. The timing analysis involved only 128 test images and a Google Colab CPU, not representative Android hardware. The results do not measure device-level memory use, energy consumption, thermal behaviour, sustained throughput, or accelerator support. They

support a relative comparison within the experimental runtime, but not a definitive mobile-deployment claim.

Class-Specific Errors and Recognition Vulnerabilities

The class-level analysis answered RQ4 by showing that the remaining errors were concentrated in a limited number of classes. EfficientNet-Lite0 achieved an F1-score of at least 0.95 in 82 classes, compared with 46 classes for MobileNetV3-Small. However, its F1-score for Lo was only 0.2353 because 13 of 15 Lo images were classified as Le. MobileNetV3-Small performed most poorly on Wo, with an F1-score of 0.2000 and nine Wo images classified as We.

Repeated directional errors included Lo to Le, Wo to We, De to Do, and Ho to He. Their recurrence suggests that the models had difficulty distinguishing specific visual structures rather than producing errors uniformly across the character inventory. Earlier Lontara studies similarly showed that recognition depends on the representation of shape, direction, segmentation, and localized character information [7], [8]. Research on other historical character datasets also identifies visually confusable classes and handwriting variation as persistent barriers in large-class recognition tasks ([9].

However, the present results do not establish the precise visual cause of each confusion. The experiment did not quantify stroke geometry, diacritic position, image quality, or writer identity. It is therefore more accurate to conclude that the errors were systematic and concentrated among particular class pairs than to claim that a specific stroke or diacritic definitively caused each error.

Overall, the findings show that deployment-oriented model selection cannot be based on one metric. EfficientNet-Lite0 provided stronger recognition performance and broader threshold-specific coverage, while MobileNetV3-Small offered substantial advantages in file size and preliminary inference speed. The preferred pipeline therefore depends on whether the application prioritizes recognition quality, computational efficiency, or a specified balance between the two.

CONCLUSION

This study compared EfficientNet-Lite0 and MobileNetV3-Small as complete transfer-learning pipelines for recognizing 114 classes of Lontara characters. The evaluation extended beyond aggregate accuracy by incorporating class-balanced performance, paired statistical comparison, confidence calibration, selective-prediction behaviour, TensorFlow Lite conversion, preliminary inference diagnostics, and class-specific error analysis.

Under the implemented experimental conditions, EfficientNet-Lite0 produced stronger recognition performance than MobileNetV3-Small. It achieved a test accuracy of 94.31% and a macro F1-score of 0.9415, compared with 90.60% and 0.9023 for MobileNetV3-Small. McNemar's exact test and paired bootstrap confidence intervals confirmed that the observed advantage in instance-level correctness and F1 performance was statistically supported on the common test set. However, this conclusion applies to the trained pipeline instances and dataset partition used in the experiment. It should not be interpreted as evidence of universal architectural superiority because the two pipelines differed in input resolution, preprocessing, backbone accessibility, and fine-tuning strategy.

The confidence analysis showed that EfficientNet-Lite0 had a lower observed Expected Calibration Error, although the confidence interval for the between-model ECE difference included zero. Confidence-based rejection increased accepted accuracy for both pipelines while reducing coverage. EfficientNet-Lite0 generally retained a larger proportion of test observations at the evaluated thresholds, but the threshold of 0.80 should be regarded as an exploratory operating point rather than an independently validated optimum.

The deployment diagnostics revealed a clear trade-off between predictive performance and computational efficiency. EfficientNet-Lite0 provided higher recognition accuracy, whereas MobileNetV3-Small produced smaller TensorFlow Lite files and substantially faster inference in the experimental CPU environment. Float16 conversion provided the most balanced result for both pipelines by reducing model size, preserving the observed sample accuracy, and producing the

shortest mean inference time within each architecture. The findings support EfficientNet-Lite0 float16 when recognition quality is the primary requirement and MobileNetV3-Small float16 when storage and inference efficiency are more restrictive. These recommendations remain preliminary because the inference measurements were obtained from 128 test images on a Google Colab CPU rather than representative mobile devices.

Class-level analysis further demonstrated that high aggregate accuracy did not eliminate severe weaknesses in particular character groups. Errors were concentrated in directional confusion pairs such as Lo to Le, Wo to We, De to Do, and Ho to He. Both pipelines also produced incorrect predictions with confidence values above 0.90, indicating that confidence thresholding alone cannot resolve systematic character-pair confusion.

Overall, the study demonstrates that lightweight Lontara recognition systems should be evaluated through an integrated framework that considers accuracy, statistical uncertainty, calibration, selective-prediction coverage, model compression, inference efficiency, and class-level vulnerabilities. Future research should evaluate writer-independent and externally collected datasets, conduct repeated training with multiple random seeds, select confidence thresholds exclusively from validation data, test converted models on the complete test set, and benchmark latency, memory use, and energy consumption on representative Android devices. Targeted augmentation, hard-negative sampling, localized feature modelling, and specialized classifiers for recurrent confusion pairs may also improve recognition of the most vulnerable Lontara classes.

ACKNOWLEDGMENT

The acknowledgement is a form of appreciation for the contribution of an institution or an individual who is not considered as the writer for example an institution or an individual who provides the research funding (funding support) of this publication.

AUTHOR CONTRIBUTION STATEMENT

The Author Contributions Statement can be up to several sentences long and should briefly describe the tasks of individual authors. Please list only 2 initials for each author, without full stops, but separated by commas (e.g. JC, JS). In the case of two authors with the same initials, please use their middle initial to differentiate between them (e.g. REW, RSW). The Author Contributions Statement should be included at the end of the manuscript before the References.

AI DISCLOSURE STATEMENT

The author used [TOOL/SERVICE NAME] during the preparation of this work for [REASON]. After using the tool/service, the author thoroughly reviewed and edited the content as needed and takes full responsibility for the content of the publication.

The authors declare that this research was prepared, researched, written, and edited without the aid of artificial intelligence (AI) techniques.

REFERENCES

[1] M. F. Adilazuarda, M. I. Wijanarko, L. Susanto, K. Nur'aini, D. T. Wijaya, and A. F. Aji (2025). NusaAksara: A multimodal and multilingual benchmark for preserving Indonesian indigenous scripts. *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 28371–28401. https://doi.org/10.18653/v1/2025.acl-long.1377.

[2] Y. S. Baso and A. Agussalim (2021). Computerization of local language characters: An innovative model for language maintenance in South Sulawesi, Indonesia. *International Journal of Advanced Computer Science and Applications*, 12(12).

[3] M. Everson (2003). Revised final proposal for encoding the Lontara (Buginese) script in the UCS. ISO/IEC JTC1/SC2/WG2 N2633R.

[4] J. Nockels, P. Gooding, S. Ames, and M. Terras (2022). Understanding the application of handwritten text recognition technology in heritage contexts: A systematic review of Transkribus in published research. *Archival Science*, 22, 367–392. https://doi.org/10.1007/s10502-022-09397-0.

[5] L. Seaward, M. Terras, and G. Muehlberger (2019). Transforming scholarship in the archives through handwritten text recognition: Transkribus as a case study. *Journal of Documentation*, 75(5), 954–976. https://doi.org/10.1108/JD-07-2018-0114.

[6] Y. G. Ferandji, Diaraya, and A. Lawi (2018). Performance comparison of image edge detection operators for Lontara Sanskrit scripts. *In Proceedings of the 2nd East Indonesia Conference on Computer and Information Technology* (pp. 241-244). IEEE.

[7] I. S. Areni, A. I. Asry, and Indrabayu (2017). A hybrid feature extraction method for accuracy improvement in "Aksara Lontara" translation. *Journal of Computer Science*, 13(9), 393-399.

[8] A. Hidayat, I. Nurtanio, and Z. Tahir (2019). Segmentation and recognition of handwritten Lontara characters using convolutional neural network. *In 2019 International Conference on Information and Communications Technology* (pp. 157-161). IEEE.

[9] A. Jalali, S. Kavuri, and M. Lee (2021). Low-shot transfer with attention for highly imbalanced cursive character recognition. *Neural Networks*, 143, 489-499.

[10] Q. A. Al-Haija (2022). Leveraging ShuffleNet transfer learning to enhance handwritten character recognition. *Gene Expression Patterns*, 45, Article 119263.

[11] K. Mohiuddin, H. Fatima, M. A. Khan, M. A. Khaleel, O. A. Nasr, and S. Shahwar (2022). Mobile learning evolution and emerging computing paradigms: An edge-based cloud architecture for reduced latencies and quick response time. *Array*, 16, 100259. https://doi.org/10.1016/j.array.2022.100259

[12] S. Okai-Ugbaje, K. Ardzejewska, and A. Imran (2022). A mobile learning framework for higher education in resource constrained environments. *Education and Information Technologies*, 27, 11947–11969. https://doi.org/10.1007/s10639-022-11094-5

[13] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang (2019). Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8), 1738–1762. https://doi.org/10.1109/JPROC.2019.2918951

[14] A. Howard, M. Sandler, G. Chu, L. C. Chen, B. Chen, M. Tan, et al. (2019). Searching for MobileNetV3. *In Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 1314-1324). IEEE.

[15] T.-J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, M. Sandler, et al. (2018). NetAdapt: Platform-aware neural network adaptation for mobile applications. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss (Eds.), Computer Vision—ECCV 2018 (pp. 289–304). *Springer.* https://doi.org/10.1007/978-3-030-01249-6_18

[16] M. Tan and Q. V. Le (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In K. Chaudhuri and R. Salakhutdinov (Eds.*), Proceedings of the 36th International Conference on Machine Learning* (Vol. 97, pp. 6105–6114). PMLR.

[17] A. S. Sambyal, U. Niyaz, N. C. Krishnan, and D. R. Bathula (2023). Understanding calibration of deep neural networks for medical image classification. *Computer Methods and Programs in Biomedicine*, 242, Article 107816.

[18] Y. Meir, O. Tevet, E. Koresh, Y. Tzach, and I. Kanter (2024). Advanced confidence methods in deep learning. *Physica A: Statistical Mechanics and its Applications*, 641, Article 129758.

[19] T. Guilbert, O. Caelen, A. Chirita, and M. Saerens (2024). Calibration methods in imbalanced binary classification. *Annals of Mathematics and Artificial Intelligence*, 92, 1319–1352.

[20] L. Huang, J. Zhao, B. Zhu, H. Chen, and S. Vanden Broucke (2020). An experimental investigation of calibration techniques for imbalanced data. *IEEE Access*, 8, 127343–127352.

[21] J. Carse, T. Suveges, S. Hogg, E. Trucco, C. Proby, C. Fleming, et al. (2021). Robust selective classification of skin lesions with asymmetric costs. In C. H. Sudre et al. (Eds.), Uncertainty for Safe Utilization of Machine Learning in Medical Imaging, and Perinatal Imaging, Placental and Preterm Image Analysis (pp. 112-121). *Lecture Notes in Computer Science*, Vol. 12959. Springer.

[22] J. Lee, M. Yu, Y. Kwon, and T. Kim (2022). Quantune: Post-training quantization of convolutional neural networks using extreme gradient boosting for fast deployment. *Future Generation Computer Systems*, 132, 124-135.

[23] T. G. Dietterich (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7), 1895-1923.

[24] M. W. Fagerland, S. Lydersen, and P. Laake (2013). The McNemar test for binary matched-pairs data: Mid-p and asymptotic are better than exact conditional. *BMC Medical Research Methodology*, 13, Article 91. https://doi.org/10.1186/1471-2288-13-9 1

[25] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger (2017). On calibration of modern neural networks. *In Proceedings of the 34th International Conference on Machine Learning* (Vol. 70, pp. 1321–1330). Proceedings of Machine Learning Research.

[26] TensorFlow (2022). Post-training quantization. TensorFlow Model Optimization.

[27] Y. Geifman and R. El-Yaniv (2019). SelectiveNet: A deep neural network with an integrated reject option. *In Proceedings of the 36th International Conference on Machine Learning* (Vol. 97, pp. 2151–2159). Proceedings of Machine Learning Research.

[28] Arsan5623 (n.d.). Aksara Lontara [Data set]. *Kaggle*. https://www.kaggle.com/datasets/arsan5623/aksara-lontara