



RESEARCH ARTICLE

Comparative Performance Analysis of MobileNetV3, EfficientNetB0, and DenseNet121 for Android-Based Cat Skin Disease Classification

Muhammad Nuruddin^{1,*}, Maulidiansyah², Zainal Arifin²

¹Department of Informatics Engineering, Faculty of Engineering, Universitas Nurul Jadid, Probolinggo, Indonesia

²Department of Information Technology, Faculty of Engineering, Universitas Nurul Jadid, Probolinggo, Indonesia

*Corresponding author: mnuruddin22@gmail.com

ARTICLE INFO

Article history:

Received: 29 October 2026

Revised: 12 November 2026

Accepted: 23 November 2026

Published: 10 December 2026

Keywords:

Cat skin disease classification

Transfer learning

Lightweight CNN

TensorFlow Lite

Android application

ABSTRACT

Skin diseases in cats such as ringworm, scabies, and flea allergy dermatitis look visually similar, and many owners cannot reach a veterinarian quickly, which encourages self-diagnosis. This study compares three ImageNet-pretrained convolutional neural networks, MobileNetV3Small, EfficientNetB0, and DenseNet121, for four-class cat skin condition classification (Flea Allergy, Health, Ringworm, and Scabies) and deploys the most suitable model in an Android application. A collection of 1,362 cat images obtained from Roboflow Universe was augmented to 3,274 images and partitioned into 2,868 training, 270 validation, and 136 test images. Each network was used as a frozen feature extractor with a new classification head and trained under identical settings. Evaluation covered accuracy, precision, recall, F1-score, confusion matrices, benchmark inference latency, throughput, and model size after conversion to TensorFlow Lite (TFLite). EfficientNetB0 reached the highest test accuracy (91.18%), followed by DenseNet121 (89.71%) and MobileNetV3 (88.97%), although differences of one to three images fall within the sampling uncertainty of a 136-image test set. MobileNetV3 was the most efficient: its TFLite model occupied 1.96 MB and needed 5.26 ms per image in the benchmark (190.21 frames per second; hardware not recorded), compared with 8.34 MB and 36.40 ms for EfficientNetB0 and 13.90 MB and 135.24 ms for DenseNet121. MobileNetV3 was therefore embedded in an application named Meow Care, which passed all seven black-box test scenarios and reached 88.97% accuracy in on-device evaluation. For resource-constrained mobile screening, a small loss in accuracy can thus be traded for much lower latency and storage. The application is a decision-support aid and does not replace veterinary diagnosis.

I. INTRODUCTION

Skin disorders are among the most common health problems of domestic cats and can reduce their quality of life through itching, hair loss, inflammation, and secondary infection [1]. Ringworm (dermatophytosis), scabies, and flea allergy dermatitis are frequently encountered in clinical practice, and ringworm and scabies are contagious and can also affect people [2], [3]. Their early signs, namely patchy alopecia, scaling, crusting, and scratch wounds, overlap considerably, so owners without veterinary training often misidentify the condition and treat it inappropriately. Access to veterinary care is also uneven: limited owner knowledge and difficulty reaching an animal clinic delay diagnosis in many cases [3]. A fast, low-cost tool that gives an initial indication from a photograph could therefore help owners decide when to seek professional care.

Deep convolutional neural networks (CNNs) learn discriminative visual features directly from images and underpin many medical and veterinary image-classification systems [4]. When labelled data are scarce, transfer learning, which reuses weights pretrained on ImageNet [5], accelerates convergence and improves generalisation [6],[7],[8],[9]. Running such models on smartphones adds constraints on model size, memory, and latency. TensorFlow Lite (TFLite) addresses them by converting trained models into a compact format that is executed on the device without an internet connection [10], [11], and recent studies in agriculture, object detection, and radiology illustrate its practicality [12],[13],[14].

Image-based recognition of cat skin diseases has already been explored on Android. Pangestu and Ramadhani [15] trained a custom sequential CNN on 400 images and reported 78.33% accuracy. That system relies on a single model, uses limited data, and is sensitive to lighting and shooting distance, and it does not examine the trade-off between accuracy and on-device efficiency across alternative architectures.

Candidate backbones embody different design choices. MobileNetV3 combines neural architecture search, NetAdapt, squeeze-and-excitation modules, and the hard-swish activation to minimise latency on mobile processors [16], [17]. EfficientNetB0 balances network depth, width, and input resolution through compound scaling [18], [19]. DenseNet121

connects each layer to all subsequent layers to improve gradient flow and feature reuse [20], [21]. A model that is more accurate on a benchmark is not necessarily the best choice for a phone, because its parameter count and inference time may be much higher [11], yet, to the best of our knowledge, this trade-off has not been quantified for cat skin conditions.

This study therefore compares MobileNetV3, EfficientNetB0, and DenseNet121 under identical data, training, and evaluation conditions to answer two research questions. RQ1: How do the three architectures compare in classifying four cat skin conditions in terms of classification performance and inference efficiency? RQ2: Can the model selected by jointly considering accuracy and efficiency be deployed in an Android application that performs on-device inference?

The main contributions of this paper are as follows:

- A controlled comparison of three pretrained CNNs for four-class cat skin condition classification, reporting accuracy, per-class precision, recall, and F1-score, confusion matrices, and 95% confidence intervals on a common 136-image test set.
- A joint accuracy–efficiency assessment that includes latency, throughput, and file size before and after TFLite conversion, together with a transparent model-selection rationale.
- Meow Care, an Android application that runs the selected MobileNetV3 TFLite model offline and is evaluated by black-box testing and on-device confusion-matrix analysis.

The remainder of the paper reviews related work (Section II), describes the materials and methods (Section III), presents the results (Section IV), discusses the findings and limitations (Section V), and concludes (Section VI).

II. RELATED WORK

Prior work relevant to this study can be grouped into cat-specific mobile systems, transfer learning for skin images in other domains, and TFLite-based deployment.

Cat skin disease recognition. Pangestu and Ramadhani [15] collected 400 images from their own photographs, Google Images, and social media and trained a four-convolutional-layer sequential CNN with an 85:15 train–test split. The model reached 78.33% accuracy, 76.96% precision, and 74.94% recall and was embedded in an Android application, but accuracy dropped with poor lighting and unfavourable shooting distance.

Transfer learning for skin images. Hwang et al. [22] classified three dog skin conditions from multispectral images of 95 dogs, comparing InceptionV3, ResNet50, DenseNet121, and MobileNetV3Small as single and consensus models. Consensus models reached accuracies of 0.87–0.89, while single models were skewed and dense fur limited the visibility of lesions. Tarisafitri and Aljabar [23] fine-tuned MobileNetV2 on 300 human skin images in three classes and reported roughly 91% training but only 70% validation and 73% test accuracy, which points to overfitting on a small dataset; only one architecture was examined.

On-device deployment. TFLite has been used to run models locally in plant pathology, object counting, assistive navigation, and radiology [12],[13],[14], [24], and lightweight backbones such as EfficientNet-Lite have been applied to produce-grading systems on constrained hardware [25]. These works demonstrate the feasibility of offline inference but do not address cat skin conditions.

Table 1 summarises the studies most closely related to ours. In contrast to them, this work compares three architectures on the same data, reports accuracy together with latency and model size, and deploys the selected model in a working application.

Table 1. Summary of closely related studies and of the present study

Ref.	Subject and task	Model(s)	Data	Main result	Reported limitation
[15]	Cat skin diseases (4 types); image classification on Android	Custom sequential CNN	400 images (own photos, web, social media); 85:15 split	Accuracy 78.33%, precision 76.96%, recall 74.94%	Small dataset; sensitive to lighting and shooting distance
[22]	Dog skin diseases (3 types); multispectral images	InceptionV3, ResNet50, DenseNet121, MobileNetV3Small; consensus models	95 dogs; augmented by rotation and translation	Consensus accuracy 0.87–0.89	Skewed single models; fur hides skin lesions
[23]	Human skin diseases (3 classes); image classification	MobileNetV2 (fine-tuned)	300 images; 80 training and 20 test per class	Training $\approx$ 91%, validation 70%, test 73% accuracy	Small dataset; large train–validation gap; single architecture
This study	Cat skin conditions (4 classes); classification and Android deployment	MobileNetV3Small, EfficientNetB0, DenseNet121 (frozen backbones)	3,274 images (2,868 / 270 / 136)	Test accuracy 88.97–91.18%; TFLite 1.96–13.90 MB, 5.26–135.24 ms	Single run; small test set (Section V)

III. MATERIALS AND METHODS

Fig. 1 gives an overview of the workflow, which proceeds from data preparation through model training and evaluation to model selection, TFLite conversion, and Android implementation.

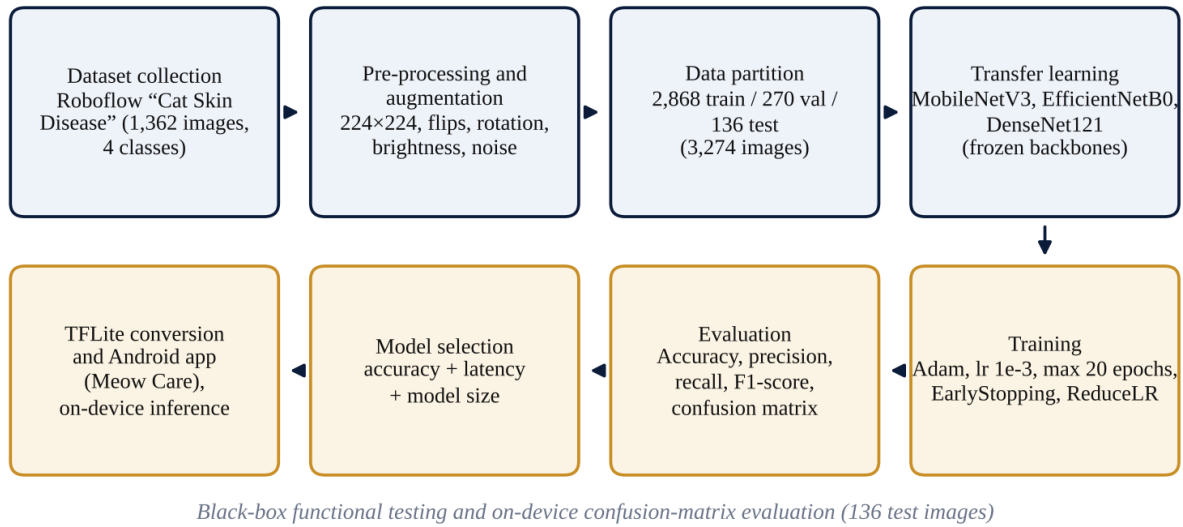


Fig. 1. Research workflow. Each of the three backbones passes through the same pipeline; the selected model is converted to TFLite and embedded in the Android application.

A. Dataset

The experiments used a collection of 1,362 cat images in four classes (Flea Allergy, Health, Ringworm, and Scabies) obtained from Roboflow Universe, where it was labelled “Cat Skin Disease” [26]. The exact source project(s), version, and licence were not recorded, and the collection may have been compiled from more than one project, so it cannot be reproduced exactly from this description. After pre-processing and augmentation on the Roboflow platform, the working dataset comprised 3,274 images, partitioned into 2,868 training images (87.60%), 270 validation images (8.25%), and 136 test images (4.15%). Table 2 shows the class distribution. The test set is moderately imbalanced (28, 28, 45, and 35 images per class), so per-class metrics are reported alongside overall accuracy.

Table 2. Class distribution of the dataset after pre-processing and augmentation

Subset	Flea Allergy	Health	Ringworm	Scabies	Total
Training	711	741	714	702	2,868
Validation	66	67	61	76	270
Test	28	28	45	35	136
Total	805	836	820	813	3,274

B. Pre-processing and Augmentation

All images were resized to 224×224 pixels, the input size of the three backbones. To increase variation in viewpoint, orientation, and illumination and to reduce overfitting, the following operations were configured in Roboflow: horizontal and vertical flips, 90° rotations (clockwise, counter-clockwise, and upside down), random rotation between -15° and $+15^\circ$, brightness variation between -25% and $+25\%$, and noise applied to up to 1.96% of the pixels. Class weights were additionally used during training to reduce the influence of class imbalance.

C. Model Architectures and Transfer Learning

Three ImageNet-pretrained backbones were loaded from *tf.keras.applications* without their original classifier: MobileNetV3Small, EfficientNetB0, and DenseNet121. In each case the backbone was frozen, so that only the newly added head was trained. The input first passed through the preprocessing function specific to each backbone. The head then consisted of global average pooling, a dropout layer (rate 0.3), a fully connected ReLU layer, a second dropout layer (0.3), and a four-unit softmax output. The fully connected layer has 128 units in the MobileNetV3 model and 256 units in the EfficientNetB0 and DenseNet121 models, as implied by the trainable parameter counts in Table 3. Because the frozen backbones account for most parameters, this strategy shortens training and limits overfitting on a small dataset, while preserving the generic edge, texture, and shape features learned from ImageNet.

Table 3. Parameter counts of the three models (frozen backbone, new classification head)

Model	Total parameters	Trainable	Frozen	Hidden units in head
MobileNetV3Small	1,013,492	74,372	939,120	128
EfficientNetB0	4,378,535	328,964	4,049,571	256
DenseNet121	7,300,932	263,428	7,037,504	256

D. Experimental Setup

The models were implemented in TensorFlow/Keras and trained on Google Colab. All three models used the same configuration (Table 4): the Adam optimiser [27] with an initial learning rate of 10^{-3} , sparse categorical cross-entropy loss, a batch size of 32, and a maximum of 20 epochs. Three callbacks controlled training: *EarlyStopping* monitored the validation loss with a patience of five epochs and restored the best weights; *ReduceLROnPlateau* multiplied the learning rate by 0.2 after three epochs without improvement in validation loss, down to 10^{-6} ; and *ModelCheckpoint* saved the model with the best validation accuracy. Each model was trained once.

Table 4. Training configuration shared by all three models

Item	Setting
Input size	$224 \times 224 \times 3$
Transfer learning	ImageNet weights; backbone frozen; new head (Section III-C)
Optimiser / learning rate	Adam / 1×10^{-3}
Loss function	Sparse categorical cross-entropy
Batch size / maximum epochs	32 / 20
Class imbalance	Class weights
EarlyStopping	Monitor: validation loss; patience: 5; restore best weights
ReduceLROnPlateau	Monitor: validation loss; factor: 0.2; patience: 3; minimum learning rate: 1×10^{-6}
ModelCheckpoint	Monitor: validation accuracy; save best only

E. Evaluation Metrics

Models were evaluated on the held-out test set using accuracy, per-class precision, recall, and F1-score, and the confusion matrix. For $C = 4$ classes and $N = 136$ test images, the metrics are defined as

$$\text{Accuracy} = \frac{1}{N} \sum_{c=1}^C \text{TP}_c \quad (1)$$

where TP_c , FP_c , and FN_c are the numbers of true positives, false positives, and false negatives of class c in a one-versus-rest view, and N is the number of test images.

$$\text{Precision}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c} \quad (2)$$

$$\text{Recall}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c} \quad (3)$$

$$\text{F1}_c = \frac{2P_c R_c}{P_c + R_c} \quad (4)$$

where P_c and R_c denote the precision and recall of class c . The macro-averaged F1-score is the unweighted mean over classes:

$$\text{Macro} - \text{F1} = \frac{1}{C} \sum_{c=1}^C \text{F1}_c \quad (5)$$

Because the test set contains only 136 images, 95% Wilson score intervals are reported for accuracy. Efficiency was measured as the mean inference latency per image (with standard deviation) over the test set, the corresponding throughput in frames per second ($\text{FPS} = 1000 / \text{latency in ms}$), and the model file size.

F. Model Selection and TFLite Conversion

The model for deployment was selected by jointly considering test accuracy, inference latency, and model size, because a model with the highest accuracy is not necessarily suited to a device with limited resources. All three trained models were converted to the TFLite format with the TensorFlow Lite converter, which also applies post-training optimisation (quantisation) to reduce file size. File sizes before and after conversion were compared, and the converted models were benchmarked on the same 136 test images for accuracy, latency, and throughput.

G. Android Implementation and Testing

The selected model was embedded in an Android application, Meow Care, developed in Android Studio with Kotlin through iterative mockup design, review, and update. The TFLite interpreter loads the model from the application's assets folder and runs inference locally with four threads. The user chooses an image from the gallery or captures one with the camera; the image is resized to 224×224 pixels with bilinear interpolation and passed to the model, and the application displays the class with the highest probability together with its confidence score. The application was tested in two ways. Black-box testing checked seven functional scenarios (home page, navigation to the detection page, gallery selection, camera capture, model prediction, result display, and the complete usage flow) and was carried out by an independent validator. On-device evaluation passed the 136 test images through the application pipeline and compared the predictions with the true labels using a confusion matrix.

IV. RESULTS

A. Training Behaviour

Fig. 2 shows the training and validation curves. MobileNetV3 improved steadily from 61.85% to 90.59% training accuracy and from 84.44% to 91.48% validation accuracy over 20 epochs. Its lowest validation loss (0.2823) occurred at epoch 17, after the learning rate had been reduced to 2×10^{-4} following epoch 11, and the weights of epoch 17 were restored. EfficientNetB0 fitted the training data fastest, reaching 97.25% training accuracy at epoch 18, where early stopping ended training. Its validation accuracy peaked at 91.85% (epochs 9, 12, and 14), but validation loss reached its minimum of 0.2479 at epoch 13 and then rose to 0.2800 while training loss kept falling; the weights of epoch 13 were restored. DenseNet121 learned more slowly (87.76% training accuracy at epoch 20) with an unchanged learning rate of 10^{-3} ; its best validation accuracy of 92.96% and lowest validation loss of 0.2186 both occurred at epoch 19, and those weights were used.

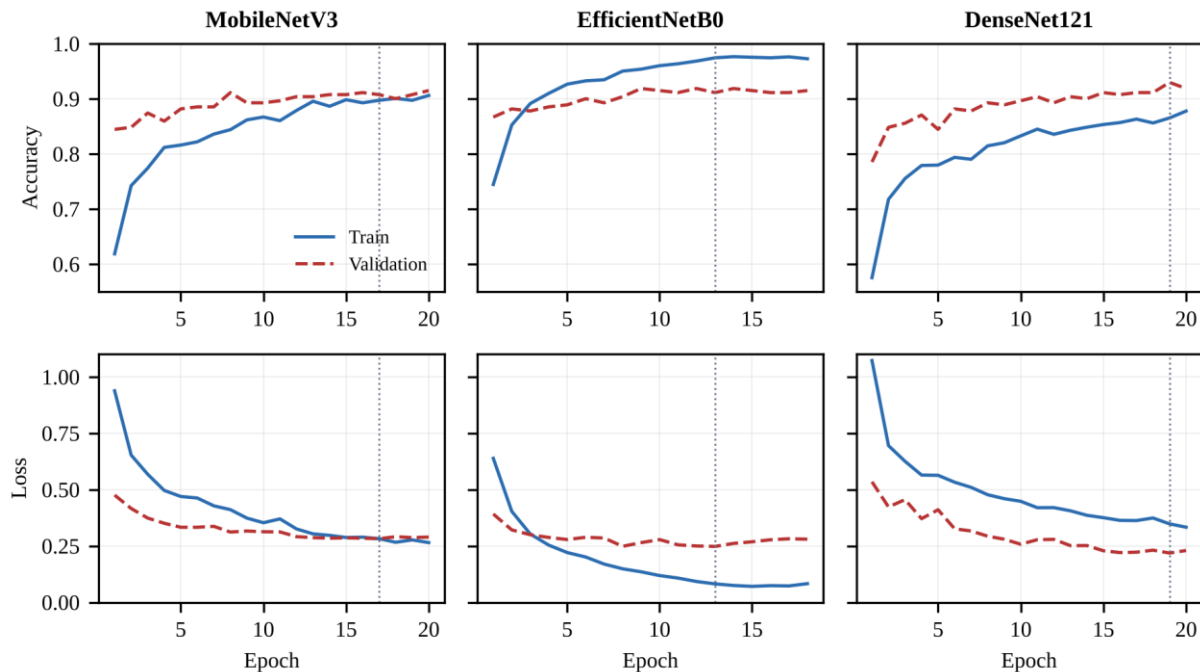


Fig. 2. Training and validation accuracy (top) and loss (bottom) of the three models. Dotted vertical lines mark the epoch whose weights were restored for evaluation.

B. Classification Performance on the Test Set

Table 5 reports per-class precision, recall, and F1-score and Fig. 3 shows the confusion matrices. EfficientNetB0 correctly classified 124 of 136 test images (91.18%), DenseNet121 122 (89.71%), and MobileNetV3 121 (88.97%); the corresponding macro-averaged F1-scores were 0.92, 0.90, and 0.89. Ringworm was the most difficult class for all models (F1-score 0.85, 0.88, and 0.84 for MobileNetV3, EfficientNetB0, and DenseNet121), and its most frequent error was prediction as Flea Allergy (6, 4, and 6 images, respectively). EfficientNetB0 and DenseNet121 never assigned a non-healthy image to the Health class (precision 1.00), but they missed three of the 28 healthy images (recall 0.89), whereas MobileNetV3 missed two (recall 0.93). Flea Allergy obtained a recall of 0.96 for every model. Because the costliest error of a screening tool is to call a sick cat healthy, the four-class results were also collapsed into a healthy-versus-diseased view derived from Fig. 3. MobileNetV3 predicted Health for two of the 108 diseased test images (sensitivity 98.1%), whereas EfficientNetB0 and DenseNet121 did so for none (100%). In the other direction, 2, 3, and 3 of the 28 healthy images were flagged as diseased (specificity 92.9%, 89.3%, and 89.3%), and the binary accuracy was 97.1%, 97.8%, and 97.8%, respectively.

Table 5. Per-class precision (P), recall (R), and F1-score on the 136-image test set

Class	Support	MobileNetV3			EfficientNetB0			DenseNet121		
		P	R	F1	P	R	F1	P	R	F1
Flea Allergy	28	0.82	0.96	0.89	0.87	0.96	0.92	0.82	0.96	0.89
Health	28	0.93	0.93	0.93	1.00	0.89	0.94	1.00	0.89	0.94
Ringworm	45	0.90	0.80	0.85	0.89	0.87	0.88	0.86	0.82	0.84
Scabies	35	0.91	0.91	0.91	0.92	0.94	0.93	0.94	0.94	0.94
Macro average	136	0.89	0.90	0.89	0.92	0.92	0.92	0.91	0.91	0.90

Note: values are computed from the confusion matrices in Fig. 3 and rounded to two decimals.

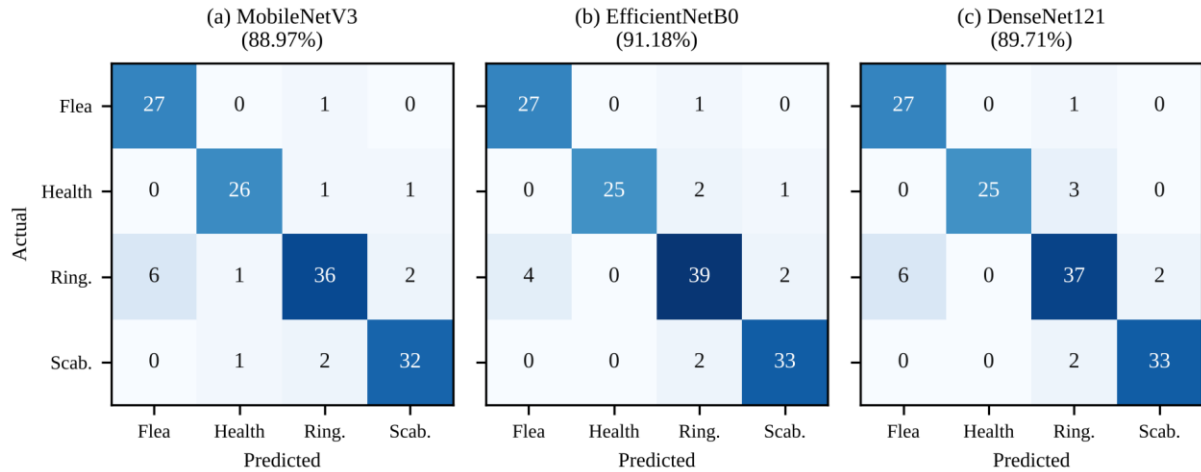


Fig. 3. Confusion matrices of (a) MobileNetV3, (b) EfficientNetB0, and (c) DenseNet121 on the 136 test images. Rows are actual classes and columns are predicted classes; the test accuracy of each model is given in parentheses.

C. Efficiency and Model Selection

Table 6 combines accuracy and efficiency. The accuracy differences are small relative to their uncertainty: the 95% intervals of the three models overlap almost completely (Table 6), and the 2.21-percentage-point gap between EfficientNetB0 and MobileNetV3 corresponds to three test images. The efficiency differences, in contrast, are large. Before conversion, mean latency per image was 104.00 ms for MobileNetV3, 172.92 ms for EfficientNetB0, and 330.93 ms for DenseNet121 (9.62, 5.78, and 3.02 FPS), and the latency variation of MobileNetV3 (standard deviation 24.76 ms) was also the smallest.

Table 6. Accuracy and efficiency of the evaluated models

Model	Accuracy, % (95% CI)	Macro-F1	Params (M)	Keras latency (ms)	TFLite size (MB)	TFLite latency (ms)	TFLite FPS	TFLite accuracy (%)
MobileNetV3	88.97 (82.6–93.2)	0.89	1.01	104.00 ± 24.76	1.96	5.26	190.21	88.97
EfficientNetB0	91.18 (85.2–94.9)	0.92	4.38	172.92 ± 55.06	8.34	36.40	27.47	91.18
DenseNet121	89.71 (83.5–93.8)	0.90	7.30	330.93 ± 127.43	13.90	135.24	7.39	90.44

Note: the best value in each column is shown in bold. Accuracy, macro-F1, and the Wilson 95% interval (CI) are computed from the test confusion matrices ($N = 136$); Keras latency is the mean ± standard deviation per image; TFLite accuracy is taken from the TFLite benchmark.

Conversion to TFLite reduced the file size of all models by 55.62% to 60.76% (Fig. 4b): MobileNetV3 shrank from 4.99 MB to 1.96 MB, EfficientNetB0 from 20.03 MB to 8.34 MB, and DenseNet121 from 31.32 MB to 13.90 MB. The converted models were also faster (Fig. 4a): mean latency fell to 5.26 ms for MobileNetV3 (190.21 FPS), 36.40 ms for EfficientNetB0 (27.47 FPS), and 135.24 ms for DenseNet121 (7.39 FPS). TFLite test accuracies were 88.97%, 91.18%, and 90.44%, respectively; the TFLite figure for DenseNet121 is one image higher than the 89.71% obtained from its Keras confusion matrix, a discrepancy that was not investigated further. Relative to EfficientNetB0, the MobileNetV3 TFLite model is 4.3 times smaller and 6.9 times faster; relative to DenseNet121 it is 7.1 times smaller and 25.7 times faster. MobileNetV3 was therefore selected for deployment.

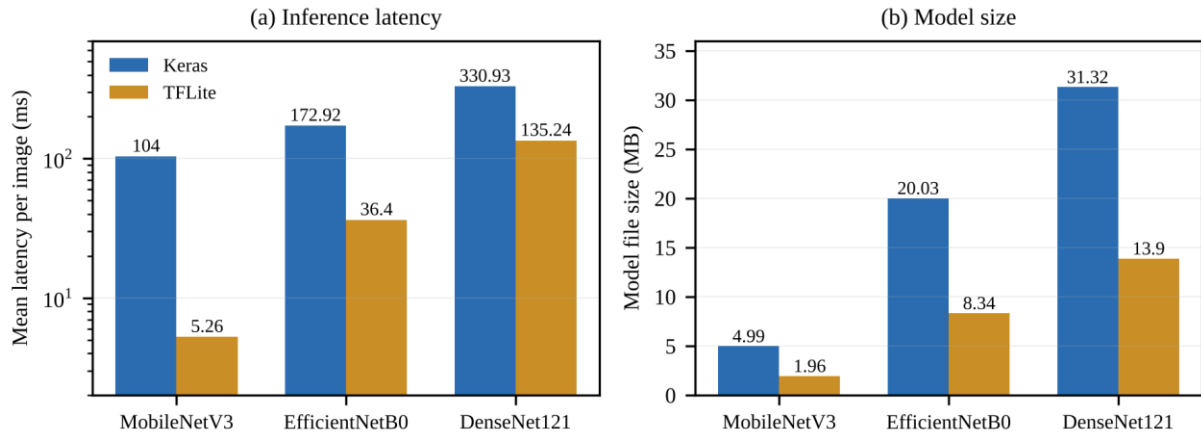


Fig. 4. Effect of TFLite conversion on (a) mean inference latency per image (logarithmic scale) and (b) model file size. The Keras and TFLite latencies were obtained with different runtimes and are therefore indicative.

D. Android Application and On-Device Evaluation

Fig. 5 shows the main screens of Meow Care. All seven black-box scenarios produced the expected output. In the on-device evaluation, the MobileNetV3 TFLite model classified 121 of the 136 test images correctly (accuracy 88.97%). The resulting confusion matrix was identical to that of the offline MobileNetV3 model (Fig. 3a), which indicates that conversion and deployment did not alter the per-class outcome on the test set.

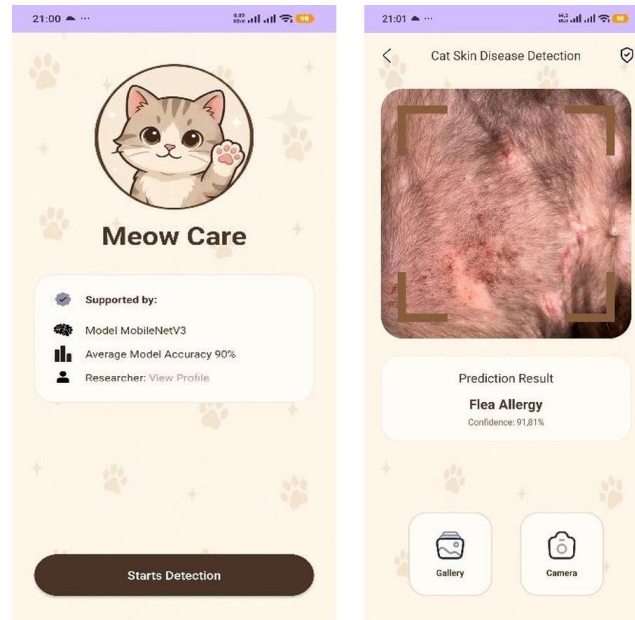


Fig. 5. Screenshots of the Meow Care application: home page (left) and detection page showing the image preview, the predicted class, and its confidence score (right).

V. DISCUSSION

Accuracy versus efficiency. The three architectures reached similar test accuracy (88.97–91.18%) but differed by a factor of 25 in TFLite latency and by a factor of seven in file size. Because a 136-image test set resolves differences of only a few images, the ranking EfficientNetB0 > DenseNet121 > MobileNetV3 should be read as a tendency rather than as a statistically established difference; the overlapping 95% intervals (Table 6) support this reading. Under this uncertainty, the large efficiency advantage of MobileNetV3 justifies its selection, which is consistent with its design for mobile processors through architecture search, squeeze-and-excitation modules, and hard-swish activation [16]. The finding agrees with the general observation that compact architectures can retain most of the accuracy of larger models at a fraction of the cost [11].

Overfitting and generalisation. EfficientNetB0 fitted the training set most closely (97.25% training accuracy) and showed the first signs of overfitting, since its validation loss increased after epoch 13 while training loss continued to fall. DenseNet121 and MobileNetV3, whose validation accuracy was equal to or higher than their training accuracy at the end of training, show no such gap. A possible explanation is that stronger fitting of a larger feature space on 2,868 augmented images leads to memorisation, while the dropout layers and augmentation reduce training accuracy relative to validation accuracy in the lighter models; this explanation was not tested here.

Error patterns. Ringworm was misclassified as Flea Allergy in 4–6 of 45 images by every model, and Health images were occasionally assigned to Ringworm. The consistency across architectures suggests that the confusion stems from the data rather than from a particular network: hair loss, scaling, and crusting occur in several of these conditions and overlap visually, which is the same difficulty that motivates automated support in the first place [2], [3]. Because the application reports a confidence score, low-confidence predictions can be flagged to the user as uncertain, a measure that should be evaluated in future work.

Comparison with earlier studies. The accuracies obtained here (89–91%) exceed the 78.33% reported for a custom CNN trained on 400 images [15] and the 73% test accuracy of a MobileNetV2 model on human skin images [23], and they are comparable to the 0.87–0.89 accuracy of consensus models for dog skin images [22]. These studies differ in subject, dataset, and protocol, so the figures are not directly comparable; the gain should not be attributed to the architecture alone. The present study contributes instead a like-for-like comparison of three backbones with efficiency measurements and an end-to-end deployment, which the cited cat-specific study [15] did not provide.

Safety-relevant errors. The healthy-versus-diseased view shows that MobileNetV3, the model selected for deployment, is the only one that called any diseased cat healthy (2 of 108 images), while the two other models erred only by over-alerting. With so few errors this difference cannot be established statistically, but it is the one measured respect in which the selected model is weaker, and it supports the confidence-aware advice and veterinary confirmation recommended below. A deployment that weights missed disease more heavily than a false alarm might therefore prefer EfficientNetB0 despite its larger size.

Implications. For developers of mobile veterinary screening tools, the results suggest selecting the backbone by the accuracy–latency–size trade-off rather than by accuracy alone. For owners, an offline application that needs no network connection and, judging from the benchmark, can respond quickly (the latency on a physical phone was not measured) can support an early, informed decision about seeking veterinary care. The application must remain a decision-support aid; clinical confirmation, for example by skin scraping or laboratory examination, is still required.

Limitations. First, the test set is small (136 images) and every model was trained once, so no variance across random seeds is available; paired significance tests such as McNemar’s test could not be run because per-image predictions were not retained. The test set was also used to choose the deployed model, so the accuracy reported for that model is not fully unbiased. Second, the images come from Roboflow Universe without a recorded project, version, or licence, and may have been compiled from several sources; label verification by veterinarians and the number of source animals are not documented, and no external validation set was used. Third, the partition was produced after augmentation on the Roboflow platform. The numbers are consistent with augmentation having been applied to the training subset only ($956 \text{ original training images} \times 3 = 2,868$, and $956 + 270 + 136 = 1,362$), but this was not audited, and leakage of augmented copies into the evaluation subsets cannot be fully excluded. Fourth, the hardware used for the latency benchmark and the specification of the Android device are not recorded, and the Keras and TFLite latencies come from different runtimes, so they should be read as relative rather than absolute values. Fifth, the on-device evaluation reused the offline test images instead of independent photographs taken in the field, and the quantisation mode was neither recorded nor varied. Finally, only frozen-backbone transfer learning was studied; fine-tuning, additional lightweight models, and more disease classes remain unexplored.

VI. CONCLUSION

This study compared MobileNetV3, EfficientNetB0, and DenseNet121 for classifying four cat skin conditions and deployed the most suitable model in an Android application. EfficientNetB0 obtained the highest test accuracy (91.18%), followed by DenseNet121 (89.71%) and MobileNetV3 (88.97%), but the differences are within the uncertainty of the 136-image test set. MobileNetV3 offered by far the best efficiency after TFLite conversion (1.96 MB, 5.26 ms per image, 190.21 FPS in a benchmark whose hardware was not recorded) and was therefore embedded in the Meow Care application, which passed all black-box scenarios and reached 88.97% on-device accuracy. In answer to the research questions, the three architectures are close in accuracy but differ greatly in efficiency, and an accuracy–efficiency trade-off allows an offline mobile classifier to be deployed with only a small loss of accuracy.

Future work should enlarge and diversify the dataset with veterinarian-verified labels and independent field photographs, repeat the experiments with several random seeds, explore fine-tuning and other lightweight architectures such as EfficientNet-Lite, benchmark latency and energy use on a range of physical Android devices, compare quantisation modes, and extend the system to more disease categories and to confidence-aware advice for the user.

ACKNOWLEDGMENT

The authors thank Lutfi Nurul Huda., for validating the black-box testing of the application. This research received no external funding and was self-funded by the authors.

AUTHOR CONTRIBUTIONS

Muhammad Nuruddin: Conceptualization, Methodology, Software, Investigation, Writing – original draft. Maulidiansyah and Zainal Arifin: Supervision, Validation, Writing – review & editing.

DATA AVAILABILITY

The images were obtained from Roboflow Universe [26]. Because the exact source projects and licences were not recorded, the image collection cannot be redistributed or reproduced exactly. The trained models and the source code are no longer available, and the results reported here are therefore documented in full in the manuscript but cannot be re-run.

DECLARATION OF COMPETING INTEREST

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

REFERENCES

- [1] L. Gunawan, S. Indarjulianto, Y. Yanuartono, R. W. Nurcahyo, and J. Prastowo, "Infestasi ektoparasit pada pasien kucing yang memiliki masalah kulit di Klinik Hewan Lilipoet Yogyakarta," *J. Sain Vet.*, vol. 42, no. 2, p. 169, 2024, doi: 10.22146/jsv.75941.
- [2] D. L. Patabang, M. S. Anthara, and I. W. Batan, "Laporan kasus: Dermatitis akibat infeksi jamur *Curvularia* sp. pada kucing peranakan Persia," *Indones. Med. Vet.*, vol. 12, no. 4, pp. 563–575, 2023, doi: 10.19087/imv.2023.12.4.563.
- [3] A. R. Permatasari, S. A. Sijid, and N. Nurmawanti, "Identifikasi dan penanganan kasus skabies pada kucing domestik (*Felis domesticus*) di UPTD Pusat Kesehatan Hewan Kota Makassar," *Filogeni: J. Mhs. Biol.*, vol. 5, no. 1, pp. 9–18, 2025, doi: 10.24252/filogeni.v5i1.53471.
- [4] T. Talaei Khoei, H. Ould Slimane, and N. Kaabouch, "Deep learning: Systematic review, models, challenges, and research directions," *Neural Comput. Appl.*, vol. 35, no. 31, pp. 23103–23124, 2023.
- [5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Miami, FL, USA, 2009, pp. 248–255, doi: 10.1109/CVPR.2009.5206848.
- [6] H. E. Kim, A. Cosa-Linan, N. Santhanam, M. Jannesari, M. E. Maros, and T. Ganslandt, "Transfer learning for medical image classification: A literature review," *BMC Med. Imaging*, vol. 22, no. 1, pp. 1–13, 2022, doi: 10.1186/s12880-022-00793-7.
- [7] C. Srinivas et al., "Deep transfer learning approaches in performance analysis of brain tumor classification using MRI images," *J. Healthc. Eng.*, vol. 2022, no. 1, Art. no. 3264367, 2022.
- [8] M. Humayun, R. Sujatha, S. N. Almuayqil, and N. Z. Jhanjhi, "A transfer learning approach with a convolutional neural network for the classification of lung carcinoma," *Healthcare*, vol. 10, no. 6, Art. no. 1058, 2022.
- [9] K. Weimann and T. O. F. Conrad, "Transfer learning for ECG classification," *Sci. Rep.*, vol. 11, no. 1, pp. 1–12, 2021, doi: 10.1038/s41598-021-84374-8.
- [10] G. Demosthenous and V. Vassiliades, "Continual learning on the edge with TensorFlow Lite," 2021, *arXiv:2105.01946*.
- [11] G. Menghani, "Efficient deep learning: A survey on making deep learning models smaller, faster, and better," *ACM Comput. Surv.*, vol. 55, no. 12, 2023, doi: 10.1145/3578938.
- [12] A. Patel et al., "Utilizing TFLite and machine learning for the early detection of mango leaf disease: An automated Flutter application," in *Proc. 5th Int. Conf. Emerg. Technol. (INCET)*, 2024, pp. 1–6.
- [13] A. Evan, M. Sarosa, R. A. Asmara, M. Kusumawardani, D. F. Al Riza, and Y. M. Azis, "Detection and counting of grape leaves using YOLOv8 via TFLite on mobile applications," in *Proc. Int. Conf. Electr. Inf. Technol. (IEIT)*, 2024, pp. 246–251.
- [14] G. Prasanna Priya, D. Bhanumurthy, P. Krishna Vamsi, L. Boni, and P. Venkata Kusuma, "Smartphone-based offline AI diagnosis of pneumothorax and hemothorax using MobileNetV3 and TFLite quantization," 2026, *SSRN 6589718*.
- [15] I. Y. Pangestu and S. R. Ramadhani, "Perancangan sistem deteksi penyakit kulit pada kucing menggunakan deep learning berbasis Android," *Teknika*, vol. 12, no. 3, pp. 173–182, 2023, doi: 10.34148/teknika.v12i3.673.
- [16] A. Howard et al., "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, South Korea, 2019, pp. 1314–1324, doi: 10.1109/ICCV.2019.00140.
- [17] L. Zhao and L. Wang, "A new lightweight network based on MobileNetV3," *KSII Trans. Internet Inf. Syst.*, vol. 16, no. 1, pp. 1–15, 2022, doi: 10.3837/tiis.2022.01.001.
- [18] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, Long Beach, CA, USA, 2019, pp. 6105–6114.
- [19] S. Sangeetha and B. Umarani, "MRI image-based Parkinson's disease classification using Deep Maxout fuzzy EfficientNet," *Biomed. Signal Process. Control*, vol. 103, Art. no. 107416, 2025.
- [20] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, 2017, pp. 2261–2269, doi: 10.1109/CVPR.2017.243.
- [21] T. S. Arulananth et al., "Classification of paediatric pneumonia using modified DenseNet-121 deep-learning model," *IEEE Access*, vol. 12, pp. 35716–35727, 2024.
- [22] S. Hwang, H. K. Shin, J. M. Park, B. Kwon, and M. G. Kang, "Classification of dog skin diseases using deep learning with images captured from multispectral imaging device," *Mol. Cell. Toxicol.*, vol. 18, no. 3, pp. 299–309, 2022, doi: 10.1007/s13273-022-00249-7.
- [23] N. Tarisafitri and A. Aljabar, "Classification of skin diseases using digital image processing with MobileNetV2 architecture," *Nusantara J. Artif. Intell. Inf. Syst.*, vol. 1, no. 1, pp. 35–44, 2025, doi: 10.47776/nuai.v1i1.1594.
- [24] N. Venkateswaran, P. Singh, R. Harini, and S. Rangdale, "Vision Assist Navigator: An Android-based application for visually impaired using TFLite detections, monocular distance analysis and TTS," in *Proc. 2nd Int. Conf. New Frontiers Commun., Autom., Manage. Secur. (ICCAMS)*, 2025, pp. 1–6.
- [25] M. A. C. Dionisio, I. J. D. Salazar, and C. C. Hortinela, "EfficientNet-Lite 4-based classification system for grading Philippine strawberries," in *Proc. IEEE Int. Conf. Artif. Intell. Eng. Technol. (IICAIET)*, 2024, pp. 366–371.
- [26] "Cat Skin Disease," image dataset, Roboflow Universe, Roboflow Inc. [Online]. Available: <https://universe.roboflow.com>
- [27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.