Available online at website: https://jurnal.iaii.or.id/index.php/RESTI

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 10 No. 1 (2026) 160 - 170 e-ISSN: 2580-0760

Bayesian Hyperparameter Optimization of Lightweight CNNs for Facial Dermatological Classification

Herimanto[1], Intan Rumondang Sianipar[2], Theo Samuel Dicunawi Aritonang[3], Ella Tasya Marito Silaban[4],
[1,2,3,4]Department of Informatics, Faculty of Informatics and Electrical Engineering, Institut Teknologi Del, Toba, Indonesia

[1]pardzheri@gmail.com, [2]intansianipar581@gmail.com, [3]theosamuelaritonang@gmail.com, [4]ellasilaban11@gmail.com

Abstract

Convolutional Neural Networks (CNNs) have been widely applied for skin condition classification. However, fair comparisons across lightweight architectures are often hindered by inconsistent hyperparameter settings. This study investigates the performance of two efficient CNN architectures, EfficientNetB3 and MobileNetV3, for facial dermatological classification across seven skin condition categories. To ensure optimal and comparable performance, Bayesian hyperparameter optimization was employed, alongside data augmentation to improve generalization. Experimental results show that EfficientNetB3 achieved the highest accuracy of 91.91%, outperforming MobileNetV3 at 90.44%. Beyond model comparison, this work highlights the novelty of applying Bayesian optimization to achieve fair benchmarking of lightweight CNNs under limited dataset conditions. The best-performing model was further deployed as a mobile application using TensorFlow Lite and Flutter, demonstrating its potential for real-world dermatological support.

Keywords: bayesian optimization; EfficientNetB3; image classification; mobile application; MobileNetV3

How to Cite: Herimanto, Intan Rumondang Sianipar, Theo Samuel Dicunawi Aritonang, and Ella Tasya Marito Silaban, "Bayesian Hyperparameter Optimization of Lightweight CNNs for Facial Dermatological Classification", *J. RESTI (Rekayasa Sist. Teknol. Inf.)* , vol. 10, no. 1, pp. 160 - 170, Feb. 2026.
Permalink/DOI: https://doi.org/10.29207/resti.v10i1.7020

Received: July 29, 2025
Accepted: January 28, 2026
Available Online: February 25, 2026

This is an open-access article under the CC BY 4.0 License
Published by Ikatan Ahli Informatika Indonesia

1. Introduction

The human skin, as the outermost organ of the body, serves as a protective barrier. Facial skin is notably more delicate and sensitive than other areas, making it more prone to dermatological issues [1]. Understanding facial skin types is crucial for proper skincare, and early detection can help reduce the cost, time, and effort of consulting dermatologists [2]. To address this challenge, Convolutional Neural Networks (CNNs) have emerged as powerful tools for image classification. CNNs can automatically extract relevant features through convolutional layers, recognizing patterns without manual feature engineering. They preserve spatial information and remain robust to variations in lighting, noise, and angles, making them well-suited for facial image classification [3]. A key advantage of CNNs is their support for transfer learning, allowing pretrained models to be fine-tuned on smaller datasets. This improves performance while reducing training costs. CNNs have excelled on benchmarks like ImageNet, CIFAR, and MNIST, and are supported by open-source libraries (e.g., TensorFlow, PyTorch, Keras), facilitating development and deployment across platforms [4]. This study compares two modern CNN architectures, EfficientNetB3 and MobileNetV3, for facial skin type detection. Both aim to balance accuracy and computational efficiency. EfficientNetB3 uses compound scaling to optimize depth, width, and resolution, making it effective on limited datasets and robust to variations in lighting and image angles [5]. It has also shown strong real-world performance, such as in COVID-19 classification from chest X-rays with 95.28% accuracy [6]. In contrast, MobileNetV3 is optimized for mobile devices using depthwise separable convolutions and squeeze-and-excitation modules to reduce parameters without major accuracy loss. It also integrates hard-swish activation and Neural Architecture Search (NAS) for optimal performance on low-power devices. It has been successfully applied to Parkinson's disease classification, achieving 99.34% accuracy, 98.53% sensitivity, and 97.78% specificity [7], [8]. Despite CNNs' success, many studies comparing architectures fail to account for fairness in experimental settings. For instance, Christin et al.

(2024) compared a conventional CNN to MobileNetV3, reporting 0.84 vs. 0.66 accuracy [9]. However, the comparison is skewed due to differences in model design and optimization and lacks standardized hyperparameters and data augmentation strategies. To address these gaps, this study conducts a fair (apple-to-apple) comparison between EfficientNetB3 and MobileNetV3, both designed for efficient image classification under constrained computational environments. While EfficientNetB3 represents manual compound scaling, MobileNetV3 reflects automated optimization via NAS. The goal is to assess trade-offs between accuracy and efficiency and identify contexts where one model is preferable over the other. Finally, this research includes the development of a mobile application with an intuitive and interactive UI, aiming to provide accurate facial skin classification and practical skincare recommendations ensuring real-world usability beyond academic results.

The main contributions of this study include a comparative evaluation of two lightweight CNN architectures, MobileNetV3 and EfficientNetB3, for facial dermatological classification, alongside the integration of Bayesian hyperparameter optimization to ensure a fair and optimal evaluation process and to address inconsistencies in prior studies. This research further analyzes model performance under limited dataset conditions, demonstrating that lightweight CNNs can still achieve competitive accuracy, and culminates in the deployment of the best-performing model into a mobile application, highlighting its practical potential for real-world dermatological support.

2. Methods

This research applies a quantitative methodology with an experimental approach to build and evaluate a facial skin type classification model. The primary approach used is transfer learning with a Convolutional Neural Network (CNN) architecture, focusing on a performance comparison between the EfficientNetB3 model, chosen for its excellence in achieving high accuracy through a compound scaling approach [5], [10], and the MobileNetV3 model, designed for superior computational efficiency through Neural Architecture Search (NAS) [7], [11], [12]. The research workflow is systematically designed through several main phases: dataset collection and preparation, model design and training, comparative evaluation, and implementation of the optimal model into a mobile application. The data preparation process includes the acquisition of 782 initial images, which are then replicated through augmentation techniques (rotation, flip, zoom, and brightness adjustment) to produce an effective training data volume of 11,367 images. To train the model, a reproducible fine-tuning strategy is applied using the progressive unfreezing method to empirically find the optimal layer depth, while hyperparameter optimization is performed using Bayesian Optimization [13] with the Adam optimizer [14]. Performance evaluation is conducted on the test data using accuracy, precision, recall, F1-score, and IoU metrics. The entire modeling process is executed using the TensorFlow v2.10 framework with hardware including an GPU P100 by Kaggle to ensure reproducibility. The following sub-chapters will describe each of these stages in more detail.

2.1 Data Collection and Preprocessing

This stage established the foundation of the research by compiling a high-quality dataset. The process involved data sourcing from multiple channels, followed by a rigorous, multi-step preprocessing pipeline to prepare the images for model training. Each step is detailed in the following sub-sections.

The initial dataset was compiled from two primary sources: public repositories on Kaggle and supplementary images gathered via web scraping. This dual-source approach was designed to obtain a diverse and representative dataset. The complete workflow of this collection process is illustrated in Figure 1.

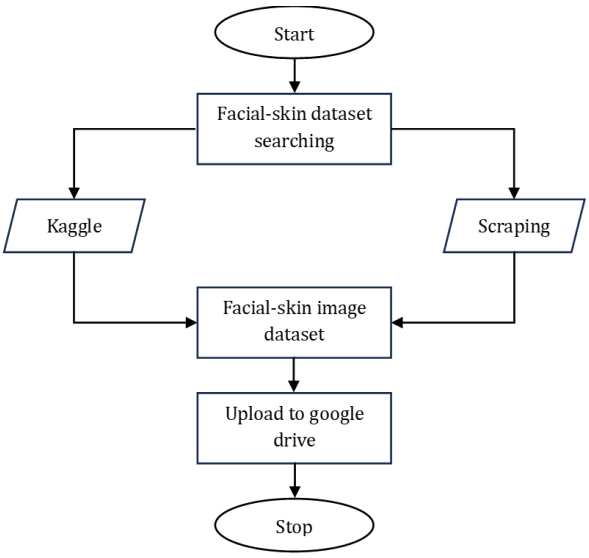


Figure 1. Data collection flow

After performing an initial data cleaning process to remove duplicates and low-quality images, a total of 782 images were curated. The distribution of this dataset across the seven skin condition classes is detailed in Table 1.

Table 1. Image Distribution by Facial Skin Category

Facial Skin Category	Number of Images
Normal	143
Acne-Prone	127
Oily	197
Dry	92
Perioral Dermatitis	60
Vitiligo	87
Aging	76

All verified data were then organized into a neat folder structure to facilitate the subsequent preprocessing steps, as shown in Figure 2. This organization aimed to

streamline the augmentation and labeling processes. Each folder was named according to its corresponding class to ensure consistency and avoid misclassification during training.

To ensure an objective model evaluation, the initial dataset was split in two stages. First, the 782 images were divided into a training set (70%) and a testing set (30%) using a stratified sampling technique to maintain class proportions. After the augmentation process, the expanded training data was then split again into a final training set (80%) and a validation set (20%) to monitor for potential overfitting during training. Next, all images underwent a preprocessing stage for standardization. The images were resized to fit the input requirements of each respective architecture: 300x300 pixels for EfficientNetB3 and 224x224 pixels for MobileNetV3. Following this, normalization was performed by converting the pixel value range from [0, 255] to [0, 1] to stabilize the training process.

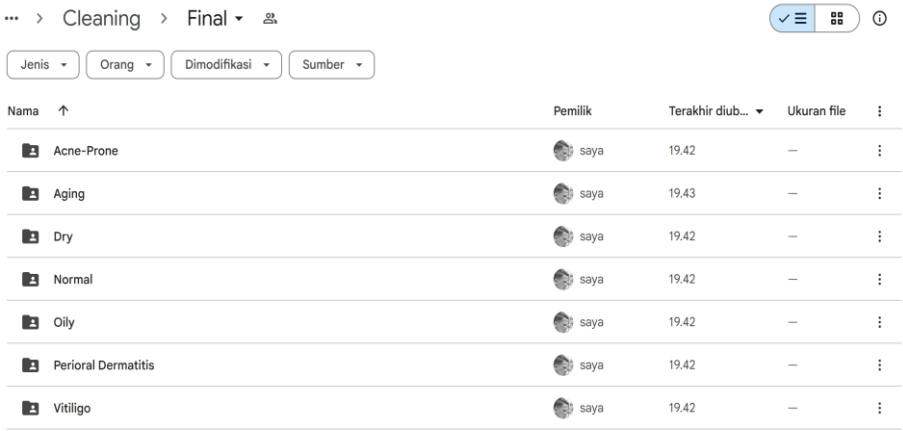


Figure 2. Folder Structure of The Dataset.

Given the initial class imbalance in the training set (e.g., 140 images for 'Oily' vs. 43 for 'Perioral Dermatitis'), a balancing strategy was implemented to prevent model bias towards majority classes. The primary approach was oversampling through targeted augmentation. Underrepresented classes, such as perioral dermatitis, vitiligo, and dry, were subjected to more intensive augmentation to synthetically increase their sample counts. This ensured the model received a more balanced exposure to each class during training, which is expected to improve performance metrics like recall for minority classes. The significant change in class distribution after this process is presented in Table 2.

Table 2. Table of Image Distribution by Facial Skin Category

Skin Category	Count Before Augmentation	Count After Augmentation
Normal	101	1537
Acne-Prone	88	1641
Oily	140	721
Dry	65	1150
Perioral Dermatitis	43	1217
Vitiligo	62	1176
Aging	48	2379
Total	547	9812

2.2 Model Design and Training Protocol

This section outlines the strategy used to train and optimize classification models. The approach includes careful architecture selection, the application of *transfer learning* with a systematic *fine-tuning* method, and a controlled training process to ensure valid and reproducible results.

EfficientNetB3 is an advanced CNN architecture designed for an optimal balance between accuracy and computational efficiency [5]. It is built on the principle of compound scaling, which systematically scales network depth, width, and resolution. For this research, the standard input resolution of 300x300 pixels was used. The core of the architecture consists of MBConv (Mobile Inverted Bottleneck Convolution) blocks, which utilize depthwise separable convolutions (with 3x3 or 5x5 kernels) for high efficiency. Each block also integrates a Squeeze-and-Excitation module to adaptively re-weight feature channels, allowing the model to focus on more relevant information. Figure 3 presents the architectural form of the EfficientNetB3 model in visual form.

MobileNetV3 is an architecture optimized for high efficiency on resource-constrained devices [7]. Its design was discovered through Neural Architecture Search (NAS), which balances latency and accuracy. This study uses the small variant, a relatively shallow network of ~57-62 layers, with a standard input resolution of 224x224 pixels. The fundamental building blocks are inverted residual blocks combined with depthwise separable convolutions to reduce computational load [12]. Key innovations include the computationally efficient h-swish activation function and the integration of Squeeze-and-Excitation modules in critical layers to improve feature sensitivity. An illustration of its core building block is shown in Figure 4.

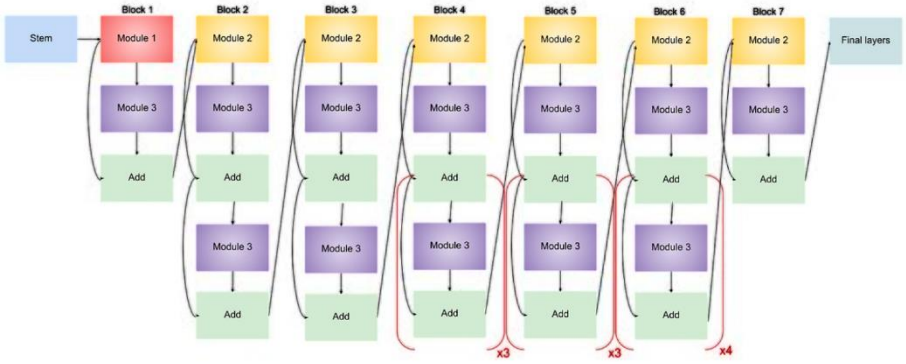


Figure 3. EfficientNetB3 Architecture [5]

The core of the training strategy was a transfer learning approach, which leverages knowledge from large-scale datasets to improve performance on the specific task of skin classification [15], [16]. Both architectures were initialized with weights pre-trained on ImageNet, providing a strong foundation of generic visual features. The adaptation of these models was performed via a systematic fine-tuning process using the progressive unfreezing method. This involved first training only the newly added classifier head while keeping the base model layers frozen. Subsequently, layers were gradually unfrozen from the top down, and the model was retrained at each stage with a lower learning rate. This controlled process allowed for the empirical identification of the optimal fine-tuning depth, the point at which the model best balanced adaptation to the new data against the risk of catastrophic forgetting. The results of this experiment, which determined the optimal number of unfrozen layers for the final training, are summarized in Table 3.

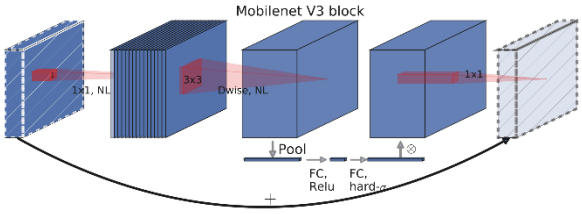


Figure 4. MobilenetV3 Block [12]

Table 3. Details of the EfficientNetB3 Layer Composition

Architecture	Fine-Tuning Configuration	Peak Validation Accuracy	Observation
EfficientNetB3	Classifier Head Only (Base Layers Frozen)	~88%	Baseline performance from generic features.
	Unfreeze ~80 Final Layers	92.09%	Optimal performance achieved.
MobileNetV3	*Classifier Head Only (Base Layers Frozen)*	85%	Baseline performance from generic features.
	Unfreeze ~70 Final Layers	90.78%	Optimal performance achieved.

To efficiently find to efficiently find the optimal hyperparameter configuration, this research applied the Bayesian Optimization approach [13]. The process was run for 20 iterations for each architecture, with the objective of maximizing validation accuracy. The explored search space included a log-uniform distribution for Learning Rate ($[10^{-5}, 10^{-3}]$) and L2 Regularization ($[10^{-6}, 10^{-3}]$), a linear range for Dropout Rate ([0.1, 0.7]), and categorical values for Batch Size ({16, 32, 48, 64}). The optimal configurations found are presented in Table 4.

All models were trained for a maximum of 50 epochs using the Adam optimizer. To prevent overfitting, an early stopping mechanism was implemented, which halted training if the validation loss did not improve for 10 consecutive epochs.

Table 4. Table of Hyperparameter Optimization Summary

Hyperparameter	Search Space	Optimal Value Found
Learning Rate	$[10^{-5}, 10^{-3}]$	0.00031
Dropout Rate	[0.1, 0.7]	0.35
L2 Regularization	$[10^{-6}, 10^{-3}]$	1.8×10^{-6}
Batch Size	{16, 32, 48, 64}	32
Peak Validation Accuracy	-	92.61%

2.3 Evaluation and Comparison

The performance of the EfficientNetB3 and MobileNetV3 architectures was quantitatively evaluated in a fair, apple-to-apple comparison on an identical, unseen test set. A comprehensive analysis was conducted using standard metrics accuracy, precision, recall, F1-score, and Intersection over Union (IoU) [17] all derived from a 7x7 confusion matrix. This multi-metric framework was designed to facilitate an in-depth analysis of each architecture's strengths and weaknesses.

All performance metrics were computed based on a 7×7 confusion matrix obtained by comparing the model's predictions on the test set with the corresponding ground truth labels. Accuracy was calculated as the ratio of correctly classified samples to the total number of samples. For each class, precision was computed as $TP/(TP + FP)$, representing the correctness of positive predictions, while recall was calculated as $TP/(TP +$

FN), indicating the model's ability to identify all relevant instances. The F1-score was derived as the harmonic mean of precision and recall using $2 \cdot (\text{Precision} \cdot \text{Recall})/(\text{Precision} + \text{Recall})$.
Additionally, Intersection over Union (IoU) for each class was calculated as $TP/(TP + FP + FN)$, measuring the overlap between predicted and true labels [17].

2.4. System Integration and Testing

The final stage involved integrating the optimal classification model into a functional and reliable application. This process included three crucial steps: converting the model to the TensorFlow Lite (TFLite) format with post-training quantization for efficient mobile deployment; developing the application with a Flutter front-end, a Supabase back-end, and an Express.js API; and validating the system's end-to-end functionality through comprehensive black-box testing [18-23].

The application adopts a modern, modular architecture with separate front-end, back-end, and API layers, as illustrated in Figure 5. The back end utilized Supabase for its real-time PostgreSQL database and user authentication services. An Express.js API [22] served as the intermediary, handling business logic and secure routing with JWT authentication.

To validate that all application features operated as specified, a black box testing methodology was employed [23]. This approach focuses on input-output validation from a user's perspective without knowledge of the internal code structure. The process was executed via manual testing [24], allowing for both scripted and exploratory validation of the user experience. A comprehensive set of over 15 test cases was designed to cover all core functionalities, including user registration, login, image classification, and history management. Both positive scenarios (e.g., a valid login) and negative scenarios (e.g., uploading a non-facial image) were tested to ensure system robustness. The testing aimed for a functional success rate of over 95% across all features.

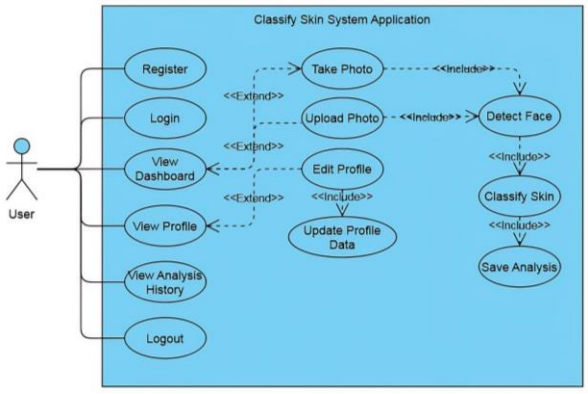


Figure 5. System Architecture Diagram

3. Results and Discussions

This section presents and discusses the results of CNN model performance and mobile application development for facial skin classification, based on experiments and real-world implementation to address the research objectives.

3.1 Results

This section presents the main findings of the study in a structured manner. It begins with the results of data collection and preprocessing, outlining the characteristics of the dataset and the preparation steps undertaken prior to model training. It then discusses the implementation and evaluation of the proposed models, including performance analysis and a comparative assessment between the two CNN architectures. Finally, this section reports the results of the application testing, where the selected model was deployed and evaluated within the developed application to assess its functionality and practical feasibility as a simulation tool.

Data collection was conducted using a combined approach from two primary sources: an open-source dataset from Kaggle and web-scraped images from reputable dermatology websites. This process aimed to obtain facial skin images with diverse dermatological conditions representative of seven categories: normal, oily, dry, acne-prone, vitiligo, aging, and perioral dermatitis. All images were manually reviewed to ensure clarity and relevance to the corresponding skin condition. Additionally, duplicates and low-quality images were removed to maintain the integrity of the dataset. The distribution of facial skin type dataset is presented in Table 5.

Table 5. Table of Image Distribution by Facial Skin Category

Facial Skin Category	Number of Images
Normal	143
Acne-Prone	127
Oily	197
Dry	92
Perioral Dermatitis	60
Vitiligo	87
Aging	76

A total of 782 images were successfully collected and underwent a manual selection and curation process to ensure visual quality and label accuracy. The initial classification was conducted through observational analysis based on the visual characteristics of each skin condition, referring to clinical dermatology literature.

The preprocessing and augmentation process was conducted using five main techniques: resizing, horizontal flip, crop (zoom), rotation, and brightness adjustment. Each technique was applied consistently across all images. However, brightness adjustment was exclusively applied to the vitiligo category, as variations in brightness help the model better capture the contrast differences characteristic of depigmented skin areas. This process resulted in a new dataset that is

ready for model training, with formats and resolutions tailored to the requirements of each model architecture. The augmented dataset also helps mitigate overfitting by increasing the diversity of training samples. More details on data preprocessing and augmentation are presented in Table 6.

Table 6. Applied Preprocessing and Augmentation Techniques

Technique	Process Description	Parameters
Resizing	Adjusting image dimensions	300x300 (EfficientNetB3) 224x224 (MobileNetV3)
Horizontal Flip	Flipping the image horizontally	Random zoom up to 20%
Crop (Zoom)	Enlarging the facial	±20 degrees
Rotation	Random image rotation	
Brightness Adjustment	Adjusting image brightness	-15% to + 15%

All preprocessed images were saved in a format and resolution compatible with the input requirements of each respective model.

The initial dataset was partitioned using an 80:20 ratio for training and testing purposes prior to augmentation. Following the augmentation process, the training data was further subdivided into training and validation subsets with a 70:30 ratio. All splitting procedures were performed while maintaining a balanced class distribution through class balancing techniques to ensure proportional representation of each label across all subsets. This approach aimed to support optimal model generalization and prevent bias toward majority classes. The validation set served as an internal benchmark for monitoring model performance during the training phase. Meanwhile, the test set was reserved exclusively for final evaluation to ensure an objective and unbiased performance assessment.

Next, the discussion will focus on the performance of the two models based on the objective values generated from the evaluation matrix.

Figure 6 illustrates the accuracy performance of the EfficientNetB3 model during the training and validation phases, demonstrating the model's learning progression across epochs. The corresponding loss curves are presented in Figure 7, providing insight into the convergence behavior and generalization stability throughout the training process.

Based on Figure 6, the training accuracy increased significantly from the beginning and reached nearly 1.00, indicating the model successfully learned the training patterns. However, slight fluctuations in the validation accuracy graph suggest some variability in the model's generalization on unseen data.

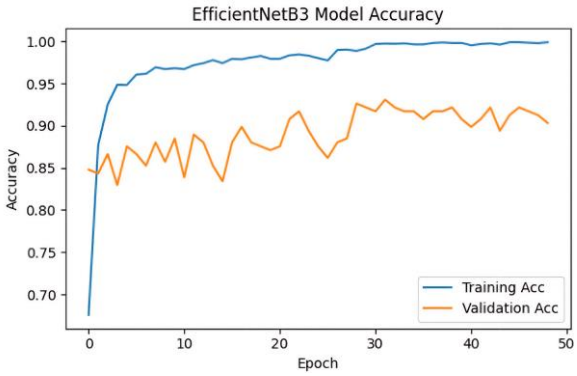


Figure 6. Accuracy Grafik EfficientNetB3 Model

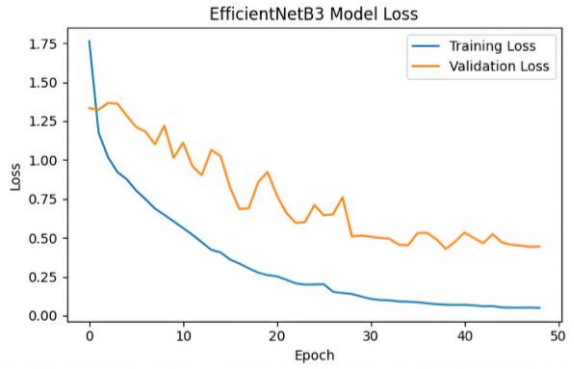


Figure 7. Loss Grafik EfficientNetB3 Model

Based on Figure 7, the training loss of EfficientNetB3 was consistently decreased to near zero, reflecting an effective learning process on the training data. However, the validation loss exhibited a more fluctuating pattern, especially after the initial epochs. The observed discrepancy between the training and validation curves of both accuracy and loss indicates a potential overfitting issue, where the model fits too closely to the training data but performs less optimally on new, unseen data. Further evaluation of classification performance is presented in the form of a confusion matrix in Figure 8.

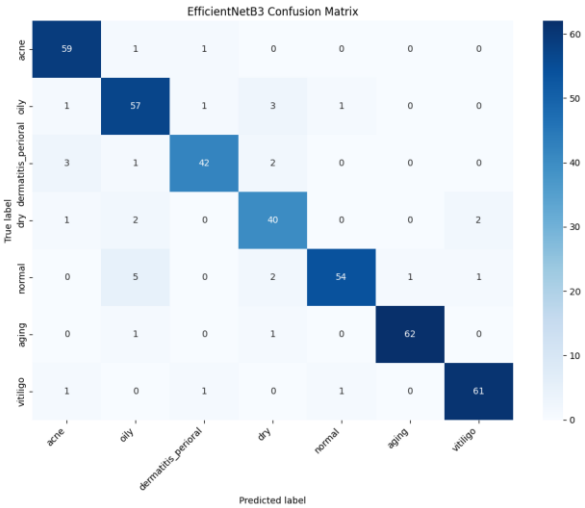


Figure 8. Confusion Matrix EfficientNetB3

The results illustrate the model's ability to classify seven facial skin type categories: acne, oily, perioral dermatitis, dry, normal, aging, and vitiligo. The diagonal values in the confusion matrix represent the number of correct predictions for each class, while the off-diagonal values indicate misclassifications. Overall, the model was able to identify most categories effectively, with many correct classifications concentrated along the diagonal. The highest number of misclassifications occurred between visually similar categories such as dry and aging, which tend to share overlapping texture features. Despite this, the model maintained high precision in dominant categories like oily and acne-prone skin. This pattern confirms the robustness of EfficientNetB3 in distinguishing subtle variations across facial skin types.

Meanwhile, the MobileNetV3 model training yielded consistently good and stable overall results, as illustrated by the accuracy and loss graphs in Figures 9 and 10.

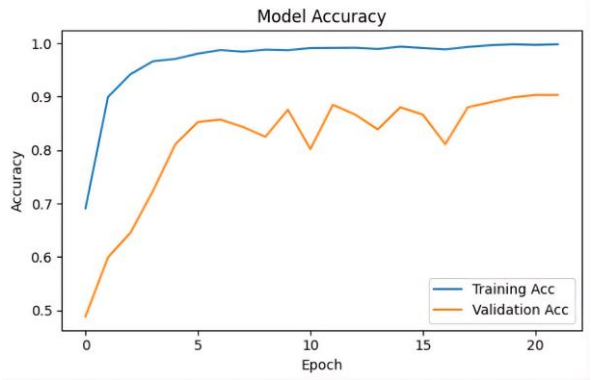


Figure 9. Accuracy Grafik MobileNetV3 Model

Based on Figure 9, the training accuracy increased significantly from the beginning of the training and reached a value close to 1.00. The validation accuracy also showed a gradual and stable increase, reaching around 0.90 by the final epoch. This indicates that the model was able to learn patterns from the training data effectively while maintaining stable performance on the validation data. The slight gap between training and validation accuracy suggests that the model generalizes well, although not perfectly. Some fluctuations can be observed in the validation accuracy curve, particularly in the middle epochs. However, the final trend shows an upward direction, indicating continuous improvement in validation performance. This outcome demonstrates that the model was not only learning efficiently but also improving its ability to classify unseen data. The consistent rise in both metrics reflects successful adaptation throughout the training process.

Figure 10 shows that the training loss dropped sharply during the early stages of training and continued to decrease toward zero. Besides that, the validation loss also followed a downward trend, although with slight fluctuations. This pattern demonstrates that the model was able to learn effectively and had a reasonably good generalization capacity on the validation data. The consistent decline in training loss confirms that the model successfully minimized error over time. Although the validation loss curve is less smooth, it does not exhibit signs of divergence or overfitting. The final stabilization of validation loss suggests that the model reached convergence after several training epochs. This stability in the loss curves indicates reliable learning dynamics. It also confirms that the training process was effective in optimizing model performance across both datasets. Further evaluation of classification performance is shown through the confusion matrix in Figure 11.

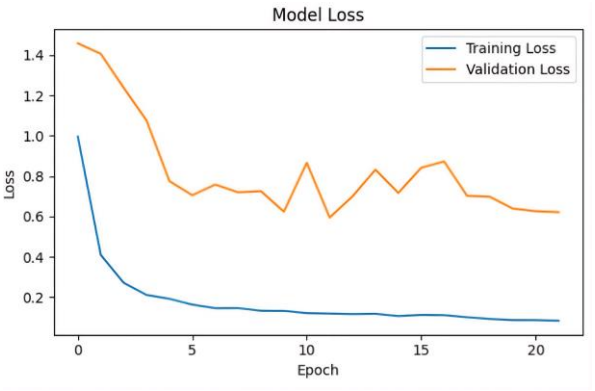


Figure 10. Loss Grafik MobileNetV3 Model

The matrix results reveal that most predictions fall along the main diagonal, indicating that the model successfully classified the majority of the data correctly. Although there are some misclassifications between classes, the overall results demonstrate that MobileNetV3 is capable of performing classification with a fairly high level of accuracy across the seven facial skin type categories.

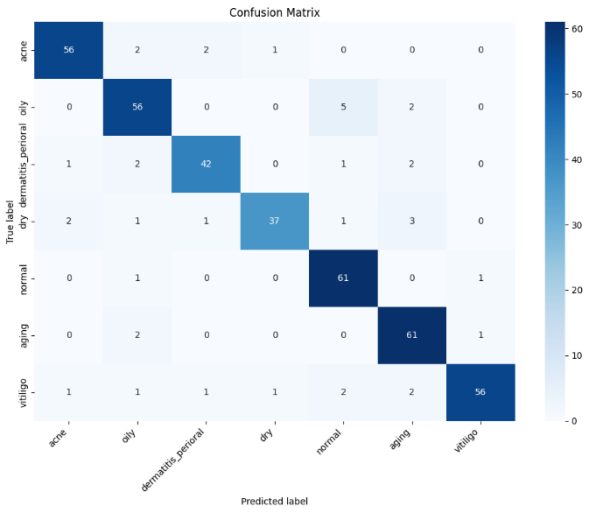


Figure 11. Confusion Matrix EfficientNetB3

The model experiment will display the number of iterations performed, accuracy, loss, learning rate, dropout value, L2 regularization value, units and batch size both MobileNetV3 and EfficientNetB3 models.

Table 7 shows the experimental results of the EfficientNetB3 model. Based on Table 7 the experimental results of the EfficientNetB3 architecture, the best performance was achieved at iteration 11, with a validation accuracy of 91.71% and a validation loss of 90.95%. This result was obtained using the following combination of hyperparameters: a learning rate of 0.000307, a dropout rate of 0.4852, an L2 regularization value of 0.0009939, 376 units, and a batch size of 13. This level of accuracy indicates that EfficientNetB3 is capable of delivering high and stable classification performance on the validation data. During the experiment, the validation loss was recorded only at iteration 1 (0.7721) and again at iteration 11 (0.9095). In all other iterations namely, iterations 2 through 10 and 12 through 20 the validation loss remained constant and was marked with the symbol "-". This condition suggests that the model's performance did not change significantly across most hyperparameter configurations tested, until an optimal combination was finally discovered at iteration 11.

Meanwhile, Table 8 shows the experimental results of the MobileNetV3 model. Based on Table 8, The best result from the MobileNetV3 experiment was obtained at the 7th iteration, with a validation accuracy of 90.78% and a validation loss of 47.52%. The combination of hyperparameters that produced this performance included a learning rate of 0.0003810794, a dropout rate of 0.2316, an L2 regularization value of 0.000250054, 309 units, and a batch size of 24. During the experiment, the validation loss was recorded only at iteration 1 (0.6265), iteration 2 (0.5518), and iteration 7 (0.4752). In all other iterations namely, iterations 3 to 6 and 8 to 20 the validation loss remained constant and was indicated by the symbol "–". This pattern indicates that the model's performance did not change significantly across most of the tested hyperparameter combinations until the optimal configuration was finally found at iteration 7.

Table 7. EfficientNetB3 Model Experiment Results

Iter	Val Acc	Val Loss	LR	DR	L2 Reg	Units	Batch Size
1	91.24%	77.21%	0.0002997	0.4852	0.0007323	376	17
2	89.4%	-	0.000301	0.2174	0.0008663	800	12
3	89.4%	-	0.000107	0.491	0.0008326	396	9
4	89.86%	-	0.0002165	0.2913	0.0005252	480	13
5	89.4%	-	0.0001838	0.2418	0.0002929	606	23
6	90.32%	-	6.246e-05	0.3331	0.0005926	376	18
7	90.78%	-	0.000274	0.3752	0.0003818	376	16
8	90.32%	-	7.915e-05	0.3752	0.0003818	374	18
9	89.86%	-	1.579e-05	0.3749	0.0004513	378	17
10	90.32%	-	0.0001687	0.4788	0.0001217	375	16
11	91.71%	90.95%	0.000307	0.4852	0.0009939	376	13
12	90.78%	-	0.0003114	0.2116	0.0005188	377	13
13	89.86%	-	0.000145	0.4959	0.000638	376	14
14	90.78%	-	0.0002039	0.4002	0.0009945	670	26
15	88.48%	-	9.558e-05	0.4212	0.000302	430	11
16	86.18%	-	0.0003738	0.3955	6.311e-05	993	15
17	88.94%	-	0.0004726	0.4099	0.0002108	375	13
18	88.94%	-	0.0001501	0.2951	0.0008188	376	12
19	88.94%	-	0.0001659	0.3535	0.0007917	964	21
20	89.4%	-	9.244e-05	0.3716	0.0004168	472	12

Table 8. MobileNetV3 Model Experiment Results

Iter	Val Acc	Val Loss	LR	DR	L2 Reg	Units	Batch Size
1	87.56%	62.65%	0.0006026719	0.4852	0.000734674	167	34
2	89.86%	55.18%	0.0006051039	0.2174	0.000867514	309	24
3	87.56%	-	0.0002202	0.491	0.0008341	174	17
4	88.02%	-	0.0004376	0.2913	0.0005295	202	25
5	88.02%	-	0.0003727	0.2418	0.0002992	244	46
6	86.18%	-	0.00009416	0.2062	0.0004016	351	17
7	90.78%	47.52%	0.0003810794	0.2316	0.000250054	309	24
8	88.48%	-	0.0008802	0.2033	0.0001642	310	25
9	89.40%	-	0.000247	0.4759	0.0009540	308	26
10	89.40%	-	0.0004645	0.3452	0.0002914	310	24
11	87.10%	-	0.0004047	0.2392	0.0004495	307	24
12	88.02%	-	0.0001715	0.2118	0.0001673	307	27
13	89.4%	-	0.000374	0.3979	0.0005382	310	25
14	84.79%	-	0.0008515	0.2712	0.0009867	309	25
15	89.4%	-	0.0001976	0.4212	0.0003083	186	23
16	89.86%	-	0.0007496	0.3955	7.155e-05	373	29
17	88.02%	-	0.0005107	0.4913	0.0008272	309	25
18	88.48%	-	0.0009353	0.3073	0.0008392	169	54
19	88.02%	-	0.0003371	0.3535	0.0007936	364	41
20	77.42%	-	0.0001914	0.3716	0.0004221	200	58

Based on the modeling results obtained from EfficientNetB3 and MobileNetV3, the following information can be drawn:

Table 9. Comparison of evaluation Results

Distinguishing parameters	EfficientNetB3	MobileNetV3
Accuracy (%)	0.9191	0.9044
Precision (%)	0.9216	0.9073
Recall (%)	0.9191	0.9044
F1-Score (%)	0.9194	0.9043

Based on Table 9, the EfficientNetB3 model demonstrated superior performance compared to MobileNetV3 across all major evaluation metrics. The highest accuracy was achieved by EfficientNetB3 at 91.91%, while MobileNetV3 obtained an accuracy of 90.44%. In terms of precision, EfficientNetB3 recorded a value of 92.16%, which was higher than MobileNetV3's score of 90.73%. Furthermore, the recall value of EfficientNetB3, at 91.91%, also exceeded that of MobileNetV3, which was 90.44%. This trend continued in the F1-score metric, where EfficientNetB3 achieved 91.94%, while MobileNetV3 reached 90.43%. These results indicate that EfficientNetB3 is more reliable and stable in classifying the various categories of facial skin types. Overall, the superior performance of EfficientNetB3 in terms of accuracy, precision, recall, and F1-score demonstrates that this model is more effective in detecting skin conditions based on the dataset used.

The trained model using TensorFlow is converted into the TensorFlow Lite (.tflite) format. This format allows the model size to be smaller and inference time to be faster without compromising the accuracy achieved during training. The conversion process is carried out using tools provided by TensorFlow, such as the TFLiteConverter. The resulting .tflite model is optimized for deployment on Android devices. This conversion is a crucial step to ensure compatibility with mobile hardware constraints and improve overall app performance.

It is important to note that the primary focus of this research is the comparative evaluation of CNN architectures for facial dermatological classification. The development of the mobile application serves as a proof-of-concept to demonstrate real-world feasibility and does not represent the central contribution of this study.

This application was developed using Flutter, an open-source framework that supports cross-platform interface development, particularly for Android and iOS. Flutter was chosen for its ability to produce responsive and consistent user interfaces across various devices. In addition, Flutter supports the integration of machine learning models in TensorFlow Lite format through the tflite_flutter plugin, making it easier to implement the model within the mobile application. The plugin provides a lightweight and efficient interface for running inference directly on the device.

The developed application includes several main features supported by a user interface (UI) design that is intuitive, user-friendly, and aesthetically pleasing. Each UI element is arranged to facilitate user interaction with the core functions of the application. In addition, the application was designed with careful attention to visual comfort and interface consistency, ensuring that the appearance is not only functional but also visually appealing.

Figure 12 displays the initial interface of the application, which consists of the landing page, sign-up, and login screens. The landing page serves as the opening screen that introduces the application's name and a brief description. The sign-up page is intended for new users to create an account, while the login page is used by registered users to securely access the application's features.

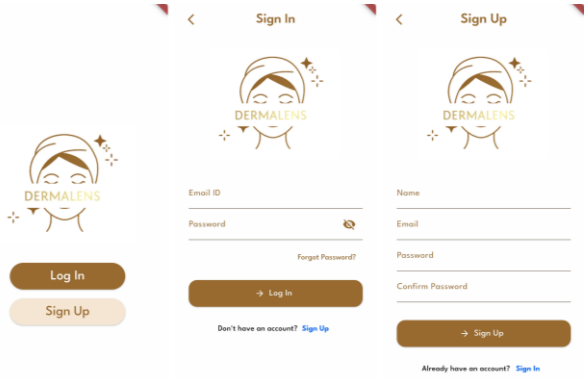


Figure 12. (a) landing page, (b) sign-up page, and (c) login page.

Figure 13 presents the dashboard and profile screens. The dashboard functions as the main navigation hub, providing access to key features such as image capture, analysis history, and account settings. The profile page allows users to view and edit their personal information, including username, email, and password. The design emphasizes clarity, with a clean layout and organized sections for easy navigation. Users are encouraged to complete their profiles to personalize their experience and enable better tracking of analysis results. Both screens were developed with consistent visual elements to ensure a seamless and cohesive user experience.

Figure 14 illustrates the core features of the application, which include the take/upload photo screen, analysis result, and analysis history. The photo capture screen enables users to take a facial image directly through the application. The classification result of the skin type is then displayed on the analysis result page. All previous analysis sessions are stored and can be reviewed on the analysis history page.

The application testing was conducted using the Black-Box Testing method, focusing on all main features. This testing aimed to ensure that each function operates according to the predefined usage scenarios. Based on the results, all features including image upload, classification process, and result display functioned

properly. The application also demonstrated good responsiveness and provided a smooth user experience on mobile devices.

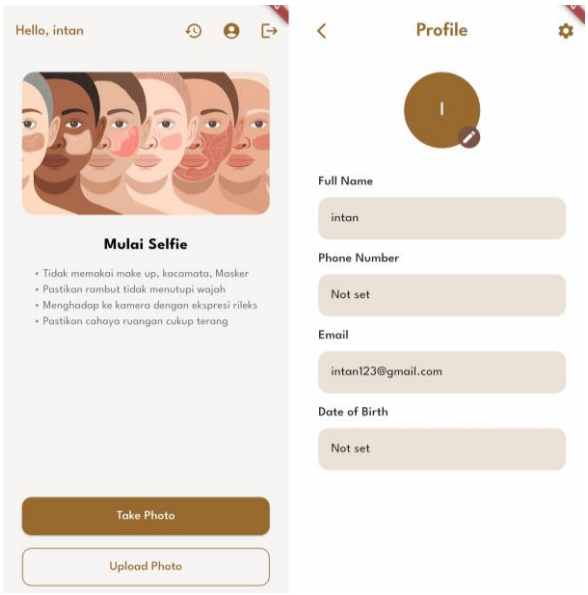


Figure 13. (a) dashboard page (b) profile page

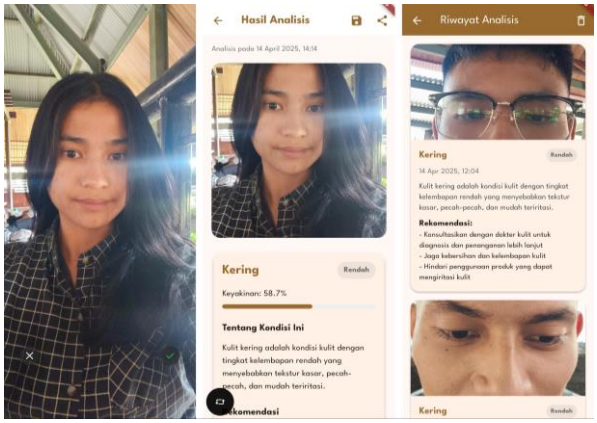


Figure 14. (a) take/upload photo page, (b) analysis result page, and (c) analysis history page.

3.2 Discussions

This study evaluated the performance of two lightweight CNN architectures, EfficientNetB3 and MobileNetV3, for facial skin type classification implemented on mobile platforms. The results showed that both models were capable of learning complex patterns from facial skin image data, albeit with different levels of effectiveness. EfficientNetB3 consistently demonstrated superior performance across all evaluation metrics. With an accuracy of 91.91%, a precision of 92.16%, a recall of 91.91%, and an F1-score of 91.94%, this model proved to be more reliable in distinguishing the seven categories of facial skin types.

In contrast, MobileNetV3, which was specifically designed for efficiency on mobile devices, achieved slightly lower results. Its best accuracy was 90.44%, with a precision of 90.73% and an F1-score of 90.43%. These results indicate that although MobileNetV3 is highly efficient, it offers slightly lower accuracy. Although there were signs of slight overfitting during the training process especially in EfficientNetB3, the application of data augmentation, class balancing, and Bayesian optimization effectively minimized this issue. The fine-tuning strategy also played a role in enhancing the model's adaptability to the dataset used. Nevertheless, one limitation of this study lies in the relatively small and less diverse dataset. Since the data collection relied primarily on Kaggle and web-scraped images, the representativeness of various demographic and dermatological variations may be limited. This may restrict the generalizability of the results, particularly when applied to broader real-world populations. Future research should address this by expanding the dataset with clinically validated and more diverse facial skin images. Furthermore, the successful integration of EfficientNetB3 into a mobile application using TensorFlow Lite and Flutter demonstrates that facial skin type classification can be effectively performed on smartphones. The quantized model maintained a high level of accuracy while significantly reducing file size and inference time, resulting in a fast and responsive user experience. Overall, these findings indicate that EfficientNetB3 is a more suitable choice for applications that prioritize classification accuracy, while MobileNetV3 remains a strong alternative for environments with limited computational resources. The integration of deep learning technology into mobile applications for dermatological analysis has the potential to assist users in identifying facial skin conditions more easily and efficiently, without requiring direct medical intervention.

4. Conclusions

This study evaluates the performance of two lightweight Convolutional Neural Network (CNN) architectures, namely EfficientNetB3 and MobileNetV3, for facial skin type classification in mobile-based applications. The evaluation was conducted using standard performance metrics such as accuracy, precision, recall, F1-score, and Intersection over Union (IoU). The results show that EfficientNetB3 outperforms MobileNetV3, achieving an accuracy of 91.73%, precision of 91.6%, recall of 91.1%, and an F1-score of 91.35%, while MobileNetV3 attained a slightly lower accuracy of 90.78%. The hyperparameter optimization process using Bayesian Optimization was successfully implemented and contributed significantly to improving the performance of both models. Based on these findings, EfficientNetB3 was selected as the optimal model and successfully deployed into a mobile application using Flutter and TensorFlow Lite, enabling accurate and practical facial skin type classification in real-world scenarios.

However, this study has limitations in terms of the availability and diversity of facial skin image datasets, as the data collection process relied primarily on

datasets from Kaggle and web scraping methods. This limitation may constrain the generalizability of the findings across diverse populations and skin conditions. Future research will focus on curating larger, clinically validated, and demographically diverse datasets to strengthen the robustness and real-world applicability of the proposed models.

Acknowledgements

The author declares no conflict of interest. This research is an original work conducted independently by the author and was not supported or influenced by any funding agency or external party.

References

[1] A. N. Mario *et al.*, "Anatomical and Physiological Characteristics of Neonatal and Infant Skin: Effects on Immunity and Dermatitis Prevalence," *JBT*, vol. 24, no. 1b, pp. 173–180, Dec. 2024, https://doi.org/10.29303/jbt.v24i1b.7918.

[2] M. A. Heryanto, D. Juananta, A. Sadanareswari, and S. A. Wulandari, "Klasifikasi Jenis Kulit Wajah menggunakan Backpropagation Neural Networks Berbasis GLCM," *ELKOMIKA*, vol. 11, no. 3, p. 705, Jul. 2023, https://doi.org/10.26760/elkomika.v11i3.705.

[3] B. Shetty, R. Fernandes, A. P. Rodrigues, R. Chengoden, S. Bhattacharya, and K. Lakshmanna, "Skin lesion classification of dermoscopic images using machine learning and convolutional neural network," *Sci Rep*, vol. 12, no. 1, p. 18134, Oct. 2022, https://doi.org/10.1038/s41598-022-22644-9.

[4] J. S. Velasco, J. V. Catipon, E. G. Monilar, V. M. Amon, G. C. Virrey, and L. K. S. Tolentino, "Classification of Skin Disease Using Transfer Learning in Convolutional Neural Networks," 2023, https://doi.org/10.48550/ARXIV.2304.02852.

[5] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 10691–10700, 2019, https://doi.org/10.48550/arXiv.1905.11946.

[6] Nadya Novalina and Mia Rizkinia, "Implementation of Xception and EfficientNetB3 for COVID-19 Detection on Chest X-Ray Image via Transfer Learning," *IJECBE*, vol. 1, no. 2, pp. 97–107, Dec. 2023, https://doi.org/10.62146/ijecbe.v1i2.14.

[7] A. Howard *et al.*, "Searching for mobileNetV3," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1314–1324, 2019, https://doi.org/10.1109/ICCV.2019.00140.

[8] S. Pechetti and B. S. Rao, "Optimized MobileNetV3: a deep learning-based Parkinson's disease classification using fused images," *PeerJ Computer Science*, vol. 9, p. e1702, Nov. 2023, https://doi.org/10.7717/peerj-cs.1702.

[9] Herimanto, Arie Satia Dharma, Junita Amalia, David Largo, and Christin Adelia Pratiwi Sihite, "Performance Analysis of MobileNetV3-based Convolutional Neural Network for Facial Skin Disorder Classification," *J. RESTI (Rekayasa Sist. Teknol. Inf.)*, vol. 8, no. 6, pp. 701–709, Dec. 2024, https://doi.org/10.29207/resti.v8i6.5982.

[10] F. Adnan, M. J. Awan, A. Mahmoud, H. Nobanee, A. Yasin, and A. M. Zain, "EfficientNetB3-Adaptive Augmented Deep Learning (AADL) for Multi-Class Plant Disease Classification," *IEEE Access*, vol. 11, pp. 85426–85440, 2023, https://doi.org/10.1109/ACCESS.2023.3303131.

[11] Q. Liu, X. She, and Q. Xia, "AI based diagnostics product design for osteosarcoma cells microscopy imaging of bone cancer patients using CA-MobileNet V3," *Journal of Bone Oncology*, vol. 49, p. 100644, Dec. 2024, https://doi.org/10.1016/j.jbo.2024.100644.

[12] K. DeVoe, G. Takahashi, E. Tarshizi, and A. Sacker, "Evaluation of the precision and accuracy in the classification of breast histopathology images using the MobileNetV3 model," *Journal of Pathology Informatics*, vol. 15, no. April, p. 100377, 2024, https://doi.org/10.1016/j.jpi.2024.100377.

[13] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016, https://doi.org/10.1109/JPROC.2015.2494218.

[14] E. Hassan, M. Y. Shams, N. A. Hikal, and S. Elmougy, "The effect of choosing optimizer algorithms to improve computer vision tasks: a comparative study," *Multimed Tools Appl*, vol. 82, no. 11, pp. 16591–16633, May 2023, https://doi.org/10.1007/s11042-022-13820-0.

[15] Y. I. Sulistya, E. T. Br Bangun, and D. A. Tyas, "CNN Ensemble Learning Method for Transfer learning: A Review," *Ilk. J. Ilm.*, vol. 15, no. 1, pp. 45–63, Apr. 2023, https://doi.org/10.33096/ilkom.v15i1.1541.45-63.

[16] A. Hosna, E. Merry, J. Gyalmo, Z. Alom, Z. Aung, and M. A. Azim, "Transfer learning: a friendly introduction," *J Big Data*, vol. 9, no. 1, p. 102, Oct. 2022, https://doi.org/10.1186/s40537-022-00652-w.

[17] M. A. Adi, Agus Khumaidi, Mohammad Basuki Rahmat, Dimas Pristovani Riananda, Muhammad Khoirul Hasin, and Didik Sukoco, "Implementasi Sistem Deteksi Titik Api Pada area graving dock Menggunakan YOLOv5," *ELK*, vol. 11, no. 2, pp. 473–482, Jul. 2024, https://doi.org/10.33795/elkolind.v11i2.5233.

[18] P. P. Ray, "A review on TinyML: State-of-the-art and prospects," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 4, pp. 1595–1623, Apr. 2022, https://doi.org/10.1016/j.jksuci.2021.11.019.

[19] M. M. Effendi, A. Turmudi, and A. Arwan, "Design of Mask Detection Application Using Tensorflow Lite based on Android Mobile," *CNAHPC*, vol. 6, no. 3, pp. 1265–1277, Jul. 2024, https://doi.org/10.47709/cnahpc.v6i3.4329.

[20] M. Szczepanik and M. Kędziora, "State Management and Software Architecture Approaches in Cross-platform Flutter Applications:," in *Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering*, Prague, Czech Republic: SCITEPRESS - Science and Technology Publications, 2020, pp. 407–414. https://doi.org/10.5220/0009411604070414.

[21] Varinia Azkarin, Rangga Gelar Guntara, and Oding Herdiana, "Development of a REST API for Human Resource Information System for Employee Referral Management Domain Using the Express JS Framework and Node.js," *JSRET*, vol. 2, no. 3, pp. 1085–1094, Jul. 2023, https://doi.org/10.58526/jsret.v2i3.199.

[22] M. Wicha and B. Pańczyk, "Performance analysis of REST API technologies using Spring and Express.js examples," *Journal of Computer Sciences Institute*, vol. 29, no. June, pp. 352–359, 2023, https://doi.org/10.35784/jcsi.3796.

[23] S. J. Putri, D. G. P. Putri, and W. H. N. Putra, "Analisis Komparasi pada Teknik Black Box Testing (Studi Kasus: Website Lars)," *Journal of Internet and Software Engineering*, vol. 5, no. 1, pp. 23–28, 2024, https://doi.org/10.22146/jise.v5i1.9446.

[24] N. Srivastava, U. Kumar, and Dr. P. Singh, "Software and Performance Testing Tools," *J. Infor. Electr. Electron. Eng.*, vol. 2, no. 1, pp. 1–12, Jan. 2021, https://doi.org/10.54060/JIEEE/002.01.001