

Classification of Avocado Varieties and Maturity Levels Using Quantized MobileNetV2 in a Mobile Application

Rahmat Hidayat[#], Hydra Amnur[#], Fajar Shidiq[#], Anla Harpanda[#]

[#]Department of Information Technology, Politeknik Negeri Padang, Padang 25164, Indonesia
E-mail: rahmat[at]pnp.ac.id, hydraamnur[at]gmail.com, fshidiq05[at]gmail.com, Anlaharpanda[at]gmail.com

ABSTRACTS

Visual grading of avocado ripeness and variety by distributors and consumers is subjective, and published deep-learning studies on avocado rarely report what happens when a trained model is integrated into a working application. This study presents a proof-of-concept, end-to-end system that classifies two Indonesian avocado varieties (Aligator and Miki) and four ripeness stages (unripe, half-ripe, ripe, overripe) from smartphone photographs. Two MobileNetV2 transfer-learning models were trained on a self-collected image set photographed from several fruits over multiple observation days, with training, validation and test subsets separated by fruit identity to prevent data leakage. The models were converted to post-training-quantized TensorFlow Lite (≈ 2.7 MB each) and served through a FastAPI–MySQL backend to a Flutter application. On a held-out test set of 32 images, the variety model classified all images correctly (100%; Wilson 95% CI 89.3–100%), and the ripeness model reached 84.4% accuracy (95% CI 68.2–93.1%), a macro-F1 of 0.814, a Cohen's κ of 0.686 and a quadratic-weighted κ of 0.900, against a majority-class baseline of 62.5% (one-sided exact binomial $p = 0.006$). Four of the five ripeness errors were half-ripe fruits predicted as unripe; half-ripe recall was 0.375 (3/8). Integration testing exposed two defects invisible to offline evaluation, a class-label ordering error and double input normalization, which caused near-constant predictions until corrected. Because the test set is small and drawn from a limited number of fruits, the results should be read as preliminary, and larger fruit-level evaluation and field testing with users are needed.

Manuscript received Jul 8, 2026;
revised Aug 9, 2026. accepted Sep
30, 2026 Date of publication Sep
30, 2026. International Journal,
JITSI : Jurnal Ilmiah Teknologi
Sistem Informasi licensed under a
Creative Commons Attribution-
Share Alike 4.0 International
License



Keywords — Avocado; Ripeness Classification; MobileNetV2; TensorFlow Lite; Post-training Quantization; Group Split; Mobile Application; ML deployment

CORRESPONDING AUTHOR

Rahmat Hidayat
Department of Information Technology, Politeknik Negeri Padang, Padang 25164, Indonesia
Email: rahmat[at]pnp.ac.id

1. INTRODUCTION

Avocado (*Persea americana* Mill.) is a high-value horticultural commodity that is prone to postharvest losses when its ripeness is misjudged at harvest, during distribution, or at purchase. Distributors and consumers usually judge ripeness by looking at the fruit and pressing it, which makes the judgement subjective and inconsistent between observers. A systematic review of technologies for avocado crops shows that computer vision and machine learning are increasingly proposed to address identification problems of this kind [1]. The task is made harder because some varieties do not show a marked change in skin colour as they ripen, so the visual cues available to a model are subtle.

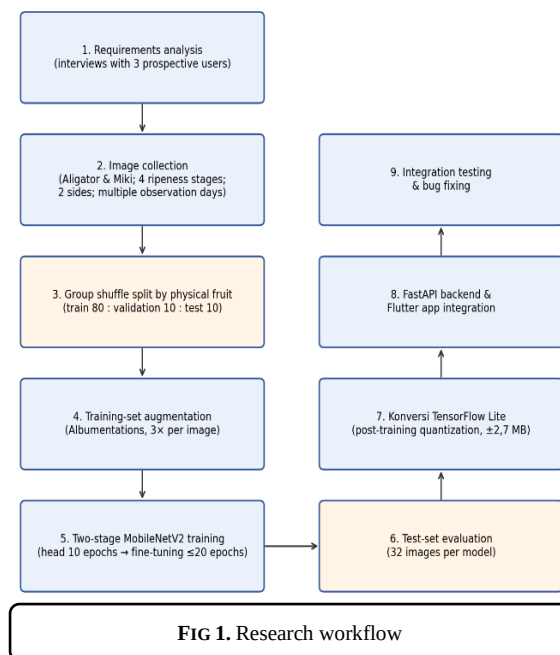
Convolutional neural networks (CNNs) are widely used for image classification in agriculture because they learn visual features directly from data [2], including for predicting fruit maturity, quality and shelf life [3]. For

avocado, Jaramillo-Acevedo et al. [4] demonstrated the feasibility of classifying Hass avocado ripeness on mobile devices using digital image processing and artificial neural networks; Xavier et al. [5] used deep learning for ripening assessment and shelf-life management of Hass avocado; Davur et al. [6] estimated Hass avocado ripeness from hyperspectral images; Nuanmeesri [7] proposed multi-channel hybrid deep neural networks for avocado ripeness classification; and Widiati and Haryanto [8] applied deep learning to classify avocado maturity in Indonesia. This literature leaves three gaps that this study addresses. First, most studies focus on Hass avocado; local Indonesian varieties such as Aligator and Miki are rarely studied, and variety identification has not been combined with ripeness classification in a single system. Second, many studies report accuracy on image-level splits, in which several images of the same fruit can fall into both the training and test sets, inflating accuracy [11]. Third, studies that stop at model evaluation rarely report whether performance is preserved once the model is quantized and served through a real service, or which engineering faults surface during integration.

To address these gaps, this study makes the following contributions: (1) an image dataset of Aligator and Miki avocados at four ripeness stages, with per-fruit metadata and a split by physical fruit identity; (2) two MobileNetV2-based classifiers [9] quantized to TensorFlow Lite [10] and evaluated with per-class metrics, confidence intervals and a comparison against a majority-class baseline; (3) an end-to-end implementation connecting the models to a FastAPI backend and a Flutter application; and (4) a report of two integration defects that offline evaluation did not detect, together with how they were detected, as practical lessons for deploying image classifiers. The study is positioned as a proof of concept; its scope is limited to the visible condition of the fruit skin and does not cover disease detection or the chemical composition of the fruit.

2. RESEARCH METHODOLOGY

The research workflow consists of requirements analysis, data collection and labelling, model training and evaluation, model conversion, system implementation, and integration testing, as shown in Fig. 1.



2.1. Requirements Analysis

Requirements were gathered through semi-structured interviews with three prospective users (one in person and two via WhatsApp) on 13–14 March 2026. Each respondent answered ten questions about how they choose avocados, their experience of choosing badly, and their expectations of a scanning application. Because the number of respondents is small, the interview findings were used as input for feature design, not as evidence that the needs generalize to all users.

2.2. Data Collection and Label Definition

Images were self-collected from several physical fruits of the Aligator and Miki varieties. Each fruit was photographed from two sides (front and back) on several observation days during ripening, so a single fruit could be recorded at more than one ripeness stage. File names encode the variety, observation day, fruit number, view and ripeness label, so every image can be traced back to its source fruit.

Ripeness labels comprise four ordinaly coded classes: 1 = unripe, 2 = half-ripe, 3 = ripe and 4 = overripe. Because a fruit can move to a different ripeness class from one day to the next, each image was labelled according to the visual condition of the fruit on the day it was photographed. Labelling in this study was not supported by objective measurements such as fruit firmness; this is discussed as a limitation in Section 3.6.

The data were split into training, validation and test sets in an 80:10:10 ratio using a group shuffle split on physical fruit identity, so that all images of a fruit, including those from different observation days, belong to only one subset. This separation prevents the model from “recognizing” the same fruit instead of learning ripeness features [11]. The class distribution of the test set, shown in Table 1, is derived directly from the confusion matrices (Fig. 3 and Fig. 4). The test set is highly imbalanced: 20 of the 32 ripeness test images (62.5%) are unripe, while the ripe and overripe classes have only two images each. Overall accuracy is therefore not sufficient to assess the model, and per-class metrics and a majority-class baseline are reported (Section 2.4).

2.3. Preprocessing, Augmentation and Model Training

Images were resized to 224×224 pixels. Pixel normalization is performed by the MobileNetV2 preprocessing layer embedded in the model, so the serving code must not normalize again (see Section 3.4).

TABLE 1. Class distribution of the test set

Model	Class	Test images	Proportion
Variety	Aligator	16	50.0%
Variety	Miki	16	50.0%
Ripeness	Unripe (1)	20	62.5%
Ripeness	Half-ripe (2)	8	25.0%
Ripeness	Ripe (3)	2	6.25%
Ripeness	Overripe (4)	2	6.25%

Augmentation was applied to the training set only, using the Albumentations library [12]: geometric transforms (rotation, horizontal/vertical flip, translation, scaling) and photometric transforms (hue-saturation-value shift, contrast, CLAHE and ISO noise to simulate smartphone camera sensors), producing three augmented images for each original training image.

The two models (variety and ripeness) were trained separately using MobileNetV2 [9] initialized with ImageNet weights. On top of the backbone, Global Average Pooling, Batch Normalization, two

Dense layers (256 and 128 units, ReLU) with Dropout of 0.4 and 0.3, and a softmax output layer were added. Training proceeded in two stages: (1) training the classifier head for 10 epochs with the backbone frozen; and (2) fine-tuning for up to 20 epochs with the first 100 backbone layers kept frozen and a reduced learning rate. Balanced class weights were applied in both stages. EarlyStopping, ReduceLROnPlateau and ModelCheckpoint monitored the validation set; the test set was not used for model or hyperparameter selection. The configuration is summarized in Table 2.

2.4. Model Evaluation

Performance was evaluated on the held-out test set using accuracy, per-class precision, recall and F1, macro-F1, balanced accuracy and Cohen's κ [13]. Because the ripeness classes are ordinal, the quadratic-weighted κ (QWK) is also reported, which penalizes errors between distant classes more heavily. 95% confidence intervals for proportions were computed with the Wilson method [14], which is more appropriate than the Wald interval for small samples. Accuracy was compared with a majority-class baseline (always predicting the most frequent class) using a one-sided exact binomial test. After evaluation, both models were converted to TensorFlow Lite with post-training quantization [10] so that they are lightweight to serve from the backend.

2.5. System Architecture and Implementation

The system has three layers (Fig. 2). The Flutter application (flutter_bloc, dio, go_router, image_picker) captures an image from the camera or gallery and sends it through a REST API. The asynchronous FastAPI backend (SQLAlchemy 2.x, aiomysql, Alembic, JWT authentication) validates the image, runs both TFLite models in sequence using the TFLite Runtime, and returns the classification result with a follow-up recommendation. A MySQL database stores user data, variety and ripeness master data, and the detection history.

The backend is deployed on Railway in a Docker container, the Android APK is built automatically through GitHub Actions, and the admin dashboard is hosted on Firebase Hosting. Inference is placed on the server so that models can be updated without redistributing the application, at the cost of requiring an internet connection. The follow-up recommendation returned to the user is practical information matched to the ripeness class, for example the estimated waiting time before the fruit is ready to eat.

2.6. System Testing

Testing covered functional testing of the application and integration testing of the model and

TABLE 2. Training configuration

Parameter	Value
Backbone/ initial weights	MobileNetV2 / ImageNet
Input size	$224 \times 224 \times 3$
Classifier head	GAP – BN – Dense 256 (ReLU, Dropout 0.4) – Dense 128 (ReLU, Dropout 0.3) – Softmax
Stage 1 (head only)	10 epochs, backbone frozen
Stage 2 (fine-tuning)	up to 20 epochs, first 100 layers frozen, reduced learning rate
Callbacks	EarlyStopping, ReduceLROnPlateau, ModelCheckpoint (validation accuracy)
Class weight	balanced (both stages)
Augmentation	Albumentations, 3 additional images per training image
Data split	Group shuffle split by fruit, 80:10:10
Software	Python, TensorFlow/Keras

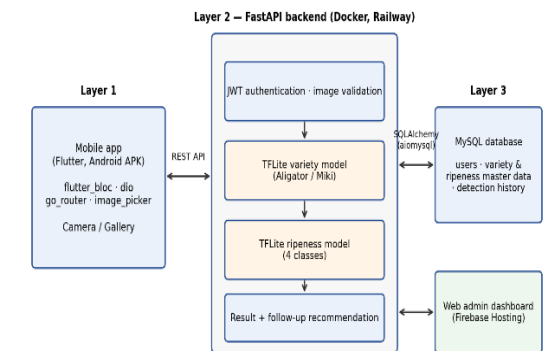


FIG 2. System architecture

backend. Integration testing used images with known labels and plain images (white and black) as a sanity check: a correctly functioning model should give different probability distributions for very different inputs. Bugs found during testing were fixed and retested

3. RESULTS AND DISCUSSION

3.1. Requirements Analysis Findings

All three respondents judged ripeness by pressing the fruit and observing skin colour and texture, and all had bought avocados that turned out to be still hard or already rotten when cut open. Respondents were also reluctant to press fruit on display shelves for fear of damaging it for other shoppers. The idea of a scanning application was received positively, with an additional wish for detecting defects or pest damage. Concerns included camera reliability, the safety of using a phone at the market, the habit of not carrying a phone while shopping, and doubts about the validity of the results. The last concern is relevant to the results in Section 3.2: the application should display the prediction confidence, especially for the half-ripe class, which is most often misclassified.

3.2. Classification Performance

The variety model correctly classified 32 of the 32 test images (Fig. 3). The Wilson 95% confidence interval for this accuracy is 89.3–100%, so with a test set of this size the true accuracy can only be said to exceed about 89%. Because the test images come from a limited number of fruits, this perfect result needs to be confirmed on more diverse fruits and imaging conditions

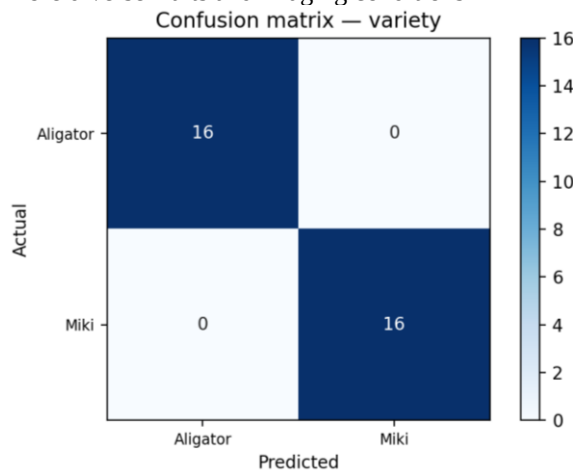


FIG 3. Confusion matrix of the variety classifier on the test set (n = 32)

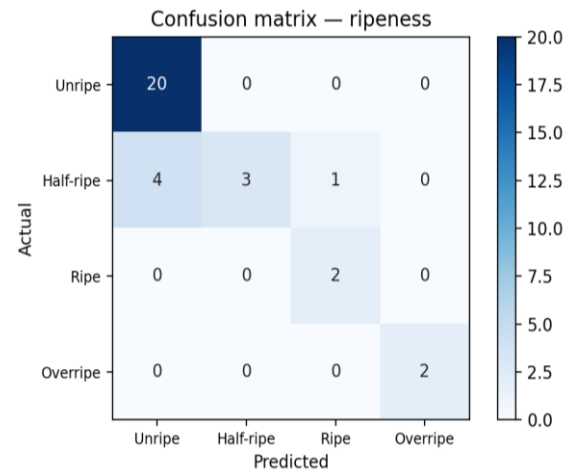


FIG 4. Confusion matrix of the ripeness classifier on the test set (n = 32)

The ripeness model correctly classified 27 of the 32 test images, an accuracy of 84.4% (95% CI 68.2–93.1%) (Fig. 4). This exceeds the majority-class baseline that always predicts “unripe” (20/32 = 62.5%; one-sided exact binomial $p = 0.006$), indicating that the model learned information beyond the class distribution. The per-class metrics in Table 3 show uneven performance: the unripe, ripe and overripe classes have a recall of 1.00, whereas the half-ripe class has only 0.375 (3/8; 95% CI 13.7–69.4%).

TABLE 3. Per-class metrics of the ripeness model (computed from Fig. 4)

Class	Support	Precision	Recall	F1
Unripe	20	0.833	1.000	0.909
Half-ripe	8	1.000	0.375	0.545
Ripe	2	0.667	1.000	0.800
Overripe	2	1.000	1.000	1.000
Macro average	32	0.875	0.844	0.814

The ripeness errors follow a systematic pattern. Four of the five errors are half-ripe fruits predicted as unripe, and one was predicted as ripe. No error skips more than one stage, which is why the QWK

TABLE 4. Summary of overall test-set metrics

Metric	Variety model	Ripeness model
Accuracy (Wilson 95% CI)	1.000 (0.893–1.000)	0.844 (0.682–0.931)
Majority-class baseline	0.500	0.625
Balanced accuracy	1.000	0.844
Macro-F1	1.000	0.814
Cohen's κ	1.000	0.686
Quadratic-weighted κ (ordinal)	–	0.900

(0.900) is much higher than the unweighted κ (0.686). In other words, the model tends to be one stage “late”: fruit that has started to ripen is still classified as unripe. A plausible explanation is that skin colour changes only subtly in the early stage of ripening, especially in varieties whose skin stays green, combined with the dominance of the unripe class in the data. This explanation remains a hypothesis to be confirmed in future work, for example by activation-map (Grad-CAM) analysis of misclassified images. For users, this error means a consumer might postpone eating fruit that is in fact nearly ready, but it does not cause unripe fruit to be recommended as ready to eat. Because the ripe and overripe classes each have only two test images, their recall of 1.00 cannot yet be considered reliable. More stable estimates require a larger test set or fruit-level cross-validation.

3.3. Model Conversion and System Implementation

Post-training quantization produced TensorFlow Lite model files of about 2.7 MB each, small enough to be loaded together by the backend. All components have been implemented and deployed: the FastAPI backend runs on Railway in a Docker container and provides interactive API documentation through Swagger UI; the TFLiteInferenceService module loads both models through the TFLite Runtime; and other modules handle image validation, detection history, and e-mail notifications for account verification and password reset. The Flutter application is distributed as an Android APK built automatically through GitHub Actions, and the admin dashboard on Firebase Hosting is used to manage variety, ripeness-stage, model and user data.

3.4. Integration Testing Findings

Integration testing of the model and backend revealed two defects that were not visible in offline evaluation. First, a label-ordering error. The inference code initially assumed that the digits 1–4 in the file names followed the alphabetical order of the class names, whereas they actually follow the natural order of ripeness (1 = unripe, 2 = half-ripe, 3 = ripe, 4 = overripe). The error was confirmed from three independent sources, namely the application reference data, the model evaluation notes, and direct testing with labelled images, and was then fixed in the inference service. Second, double normalization. Pixels were normalized in the backend code and again by the MobileNetV2 preprocessing layer inside the model, so the model output was nearly constant regardless of the image. The defect was detected when plain white and black images produced almost identical predictions, and was fixed by removing the manual normalization.

Both defects were “silent”: the service kept running and returned plausible-looking labels, so they would not have been caught by ordinary functional testing or by test-set accuracy computed in the training environment. These findings support the following practices for deploying image classifiers: (1) store the index-to-label mapping as an artefact together with the model rather than reconstructing it in the serving code; (2) document explicitly whether preprocessing is inside or outside the model; and (3) run a regression test that compares service predictions with those of the original model on a set of labelled images whenever the model or the serving code changes.

3.5. Comparison with Previous Studies

A direct comparison of accuracy figures with previous studies [4]–[8] is not possible because of differences in variety, number of classes, imaging modality (RGB or hyperspectral) and data-splitting protocol. The main differences of this study are its focus on the local Aligator and Miki varieties, the split by physical fruit, the reporting of per-class metrics with confidence intervals, and the reporting of performance after the model was integrated into a real service.

3.6. Limitations

- The test set is small (32 images per model) and highly imbalanced; the ripe and overripe classes have only two test images each, so per-class estimates are highly uncertain.
- The data come from a limited number of fruits and imaging conditions; robustness to different lighting, backgrounds and cameras has not been tested.
- Ripeness labels were assigned visually without objective measurements (e.g., fruit firmness), so some errors in the transitional class may stem from label uncertainty itself.
- Performance has not been estimated with fruit-level cross-validation, and neither the agreement between the TFLite model and the Keras model before quantization nor the system latency has been measured separately.
- Server-side inference makes the application dependent on an internet connection.
- The requirements analysis involved only three respondents, and usability testing and field testing with farmers, distributors and consumers have not yet been carried out comprehensively

4. DISCUSSION

This study built an end-to-end system for classifying the variety (Aligator and Miki) and four ripeness stages of avocado using MobileNetV2 quantized to TensorFlow Lite and served through a FastAPI backend to a Flutter application. On a 32-image test set separated by fruit identity, the variety model achieved 100% accuracy (95% CI 89.3–100%) and the ripeness model 84.4% (95% CI 68.2–93.1%; macro-F1 0.814; QWK 0.900), above the

majority-class baseline of 62.5%. The main weakness of the model is the half-ripe class (recall 0.375), which is mostly predicted as unripe. Integration testing uncovered two silent defects, label ordering and double normalization, showing that offline accuracy does not guarantee that the system is correct after deployment.

Given the size of the test set, these results are preliminary. Future work should: (1) increase the number of fruits, particularly in the half-ripe, ripe and overripe classes, with labels based on objective measures such as fruit firmness; (2) perform fruit-level cross-validation and compare the TFLite model with the model before quantization; (3) test under diverse lighting, backgrounds and devices, including usability and field testing with farmers and distributors; (4) display confidence scores and flag uncertain predictions in the application; and (5) evaluate on-device inference to reduce dependence on an internet connection.

REFERENSI

- [1] T. Ramirez-Guerrero, M. I. Hernandez-Perez, M. S. Tabares, A. Marulanda-Tobon, E. Villanueva, and A. Peña, "Agroclimatic and phytosanitary events and emerging technologies for their identification in avocado crops: A systematic literature review," *Agronomy*, vol. 13, no. 8, Art. no. 1976, 2023, doi: 10.3390/agronomy13081976.
- [2] M. Dyrmann, H. Karstoft, and H. S. Midtiby, "Plant species classification using deep convolutional neural network," *Biosystems Engineering*, vol. 151, pp. 72–80, 2016, doi: 10.1016/j.biosystemseng.2016.08.024.
- [3] N. Aherwadi, U. Mittal, J. Singla, N. Z. Jhanjhi, A. Yassine, and M. S. Hossain, "Prediction of fruit maturity, quality, and its life using deep learning algorithms," *Electronics*, vol. 11, no. 24, Art. no. 4100, 2022, doi: 10.3390/electronics11244100.
- [4] C. A. Jaramillo-Acevedo, W. E. Choque-Valderrama, G. E. Guerrero-Álvarez, and C. A. Meneses-Escobar, "Hass avocado ripeness classification by mobile devices using digital image processing and ANN methods," *International Journal of Food Engineering*, vol. 16, no. 12, 2020, doi: 10.1515/ijfe-2019-0161.
- [5] P. Xavier, P. M. Rodrigues, and C. L. M. Silva, "Shelf-life management and ripening assessment of 'Hass' avocado (*Persea americana*) using deep learning approaches," *Foods*, vol. 13, no. 8, Art. no. 1150, 2024, doi: 10.3390/foods13081150.
- [6] Hidayat, R., Nuryana, Z., Erianda, A., Alanda, A., Rasyidah, Amnur, H. (2026). Visual Analysis and Scientometric Mapping of Deep Learning Techniques for Grapevine Disease Classification. In: Badioze Zaman, H., et al. *Advances in Visual Informatics. IVIC 2025. Lecture Notes in Computer Science*, vol 16215. Springer, Singapore.
- [7] Y. J. Davur, W. Kämper, K. Khoshelham, S. J. Trueman, and S. H. Bai, "Estimating the ripeness of Hass avocado fruit using deep learning with hyperspectral imaging," *Horticulturae*, vol. 9, no. 5, Art. no. 599, 2023, doi: 10.3390/horticulturae9050599.
- [8] N. Ruseno, F. Ernawan, M. A. Salamat, M. S. Arshad, and H. Amnur, "An Enhanced YOLOv8 Model Integrated with CNN for Real-Time Vehicle License Plate Recognition," *JOIV : International Journal on Informatics Visualization*, vol. 10, no. 1, doi: <http://dx.doi.org/10.62527/joiv.10.1.5145>.
- [9] S. Nuanmeesri, "Utilization of multi-channel hybrid deep neural networks for avocado ripeness classification," *Engineering, Technology & Applied Science Research*, vol. 14, no. 4, pp. 14862–14867, 2024, doi: 10.48084/etasr.7651.
- [10] W. Widiati and T. Haryanto, "Deep learning for automatic classification of avocado fruit maturity," *Jurnal Pilar Nusa Mandiri*, vol. 20, no. 1, pp. 75–80, 2024, doi: 10.33480/pilar.v20i1.5043.
- [11] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520, doi: 10.1109/CVPR.2018.00474.
- [12] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2704–2713, doi: 10.1109/CVPR.2018.00286.
- [13] S. Kapoor and A. Narayanan, "Leakage and the reproducibility crisis in machine-learning-based science," *Patterns*, vol. 4, no. 9, Art. no. 100804, 2023, doi: 10.1016/j.patter.2023.100804.

- [14] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: Fast and flexible image augmentations," *Information*, vol. 11, no. 2, Art. no. 125, 2020, doi: 10.3390/info11020125.
- [15] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960, doi: 10.1177/001316446002000104.
- [16] L. D. Brown, T. T. Cai, and A. DasGupta, "Interval estimation for a binomial proportion," *Statistical Science*, vol. 16, no. 2, pp. 101–133, 2001, doi: 10.1214/ss/1009213286.