

**PENERAPAN *DATASET* CITRA DAUN KRATOM UNTUK
KLASIFIKASI KESEHATAN DAUN MENGGUNAKAN TEKNIK
AUGMENTASI CITRA BERBASIS *MACHINE LEARNING***

LAPORAN TUGAS AKHIR

PERPUSTAKAAN

NPP. 6171052A2000001



JESY KURNIATI

NIM: 23021080

POLITEKNIK 'AISYIYAH PONTIANAK

PROGRAM STUDI D-III TEKNOLOGI INFORMASI

POLITEKNIK 'AISYIYAH PONTIANAK

2026

**PENERAPAN *DATASET* CITRA DAUN KRATOM UNTUK
KLASIFIKASI KESEHATAN DAUN MENGGUNAKAN TEKNIK
AUGMENTASI CITRA BERBASIS *MACHINE LEARNING***

LAPORAN TUGAS AKHIR



NPP. 6171052A2000001

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Ahli Madya Komputer

JESY KURNIATI

NIM: 23021080

POLITEKNIK 'AISYIYAH PONTIANAK

**PROGRAM STUDI D-III TEKNOLOGI INFORMASI
POLITEKNIK 'AISYIYAH PONTIANAK
2026**

LEMBAR PERSETUJUAN

**PENERAPAN *DATASET* CITRA DAUN KRATOM UNTUK
KLASIFIKASI KESEHATAN DAUN MENGGUNAKAN TEKNIK
AUGMENTASI CITRA BERBASIS *MACHINE LEARNING***

LAPORAN TUGAS AKHIR

JESY KURNIATI

NIM: 23021080

Telah disetujui untuk diajukan
mengikuti Seminar Tugas Akhir di
Politeknik Aisyiyah Pontianak
Pada Senin, 6 Juli 2026

Menyetujui,
Pembimbing



Muhammad Faqih Dzulgarnain, S.T., M.Cs.

NIDN. 1128089301

LEMBAR PENGESAHAN

PENERAPAN *DATASET* CITRA DAUN KRATOM UNTUK KLASIFIKASI KESEHATAN DAUN MENGGUNAKAN TEKNIK AUGMENTASI CITRA BERBASIS *MACHINE LEARNING*

Laporan Tugas Akhir Ini Telah Disidangkan
Pada Tanggal 6 Juli 2026

001

Mengesahkan:

Ketua Penguji,

Anggota Penguji,



Hermanto, S.Kom., M.Kom

Muhammad Faqih Dzulqarnain, S.T., M.Cs.

NIDN : 0311069401

NIDN : 1128089301

Mengetahui:

Ketua Program Studi D-III Teknologi Informasi



Irwan Adhi Prasetya, S.T., M.T

NIDN : 1121079302

PERPUSTAKAAN

NPP. 6171052A2000001



POLITEKNIK 'AISYIYAH PONTIANAK

PERNYATAAN INTEGRITAS AKADEMIK

Saya yang bertanda tangan di bawah ini:

Nama : JESY KURNIATI

NIM : 23021080

Program Studi : D3 Teknologi Informasi

Dengan ini menyatakan bahwa dokumen ilmiah Tugas Akhir (LTA) dengan judul

**"PENERAPAN *DATASET* CITRA DAUN KRATOM UNTUK
KLASIFIKASI KESEHATAN DAUN MENGGUNAKAN TEKNIK
AUGMENTASI CITRA BERBASIS *MACHINE LEARNING*"**

yang ditulis ini tidak mengandung unsur pelanggaran integritas akademik sesuai Peraturan Menteri pendidikan, Kebudayaan, Riset, dan Teknologi Republik Indonesia Nomor 39 tahun 2021 tentang Integritas Akademik dalam Menghasilkan Karya Ilmiah. Apabila di kemudian hari dokumen ilmiah tugas akhir ini ditemukan terbukti mengandung unsur pelanggaran integritas akademik, maka saya bersedia menerima sanksi akademik dan sanksi hukum sesuai perundang-undangan yang berlaku.



PC

Pontianak, 6 Juli 2026

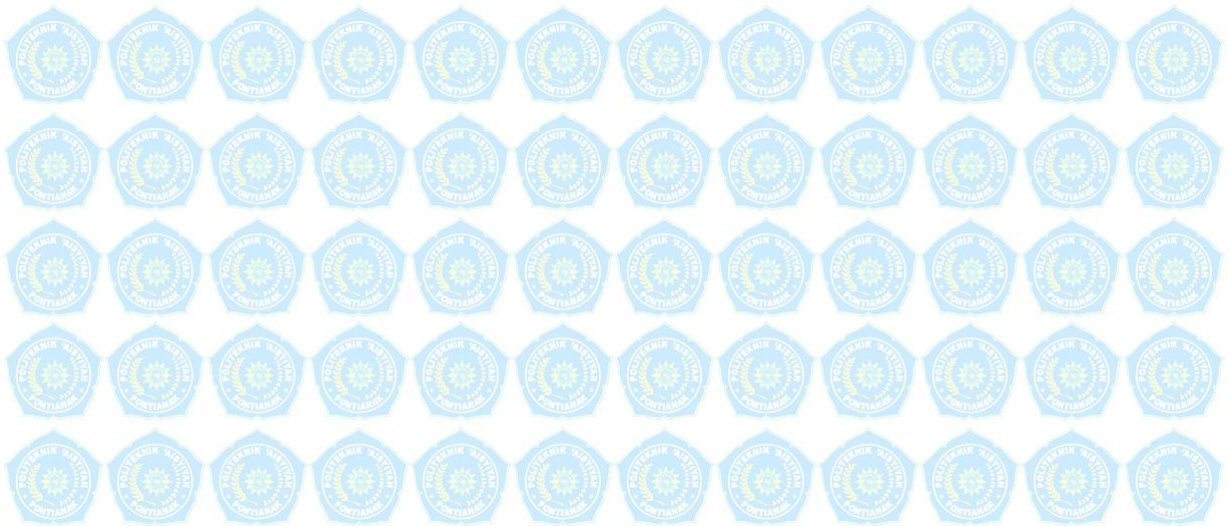


(Jesy Kurniati)

23021080

PERPUSTAKAAN

NPP. 6171052A2000001



POLITEKNIK 'AISYIYAH PONTIANAK

KATA PENGANTAR

Puji syukur peneliti panjatkan kehadiran Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya sehingga peneliti dapat menyelesaikan Laporan Tugas Akhir yang berjudul ” **PENERAPAN *DATASET* CITRA DAUN KRATOM UNTUK KLASIFIKASI KESEHATAN DAUN MENGGUNAKAN TEKNIK AUGMENTASI CITRA BERBASIS *MACHINE LEARNING***”.

Peneliti menyadari bahwa keberhasilan penyusunan Laporan Tugas Akhir ini tidak lepas dari bimbingan dan dukungan pihak-pihak yang telah membantu peneliti. Oleh karena itu, peneliti ingin menyampaikan ucapan terima kasih kepada semua pihak yang telah membantu terlaksana dan terselesaikannya Laporan Tugas Akhir ini kepada:

1. Ibu Tilawaty Aprina, S.ST., M.Kes., selaku Direktur Politeknik 'Aisyiyah Pontianak.
2. Kedua Orang Tua dan keluarga peneliti yang telah memberikan dukungan materi dan non-materi kepada peneliti.
3. Bapak Irwan Adhi Prasetya, S.T., M.T. selaku Ketua Program Studi D-III Teknologi Informasi.
4. Bapak Muhammad Faqih Dzulqarnain, S.T., M.Cs. selaku Pembimbing dan Penguji 2 yang telah memberikan bimbingan dan masukan pada penyusunan Laporan Tugas Akhir ini.
5. Bapak Hermanto S.Kom., M.Kom. selaku Ketua Penguji yang telah memberikan masukan pada penyusunan Laporan Tugas Akhir ini.
6. Rekan-rekan seangkatan Tahun 2023 yang telah menemani, memberikan dukungan, serta berbagi suka dan duka selama masa perkuliahan hingga penyusunan Laporan Tugas Akhir ini.
7. Pihak lain yang telah membantu penyusunan Laporan Tugas Akhir ini.

Penyusunan Laporan Tugas Akhir ini diharapkan dapat memberikan pengetahuan dan penerapan ilmu yang telah diperoleh di lingkungan pendidikan peneliti kepada masyarakat untuk dipraktikkan lebih lanjut. Peneliti juga menyadari bahwa penyusunan laporan ini masih terdapat ketidaksempurnaan, oleh karena itu peneliti mengharap kritik dan saran sebagai perbaikan kedepannya dari laporan ini.

Pontianak, 6 Juli 2026

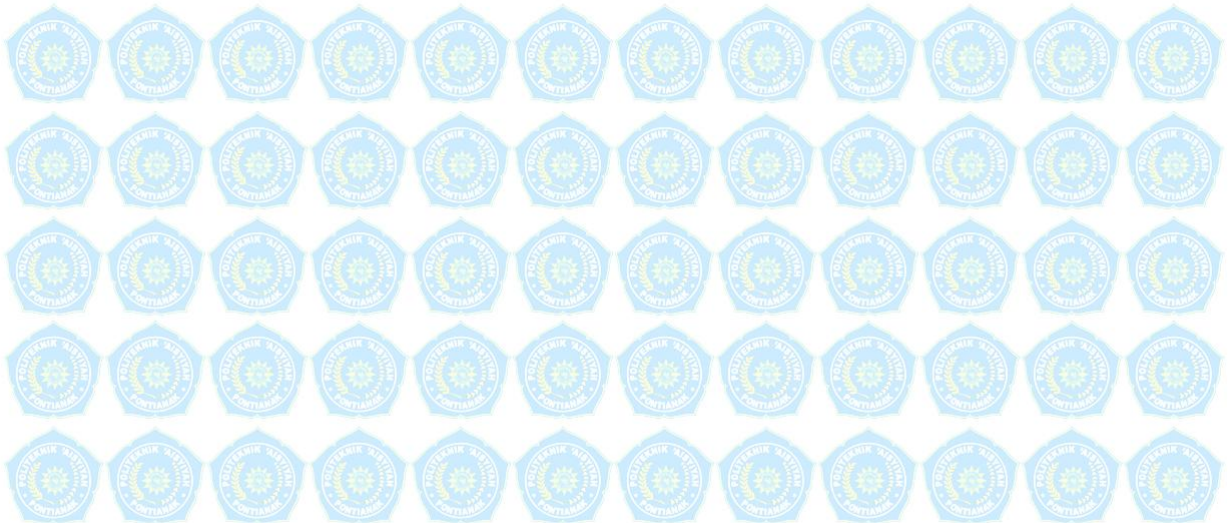
Peneliti



Yesy Kurniati

PERPUSTAKAAN

NPP. 6171052A2000001



POLITEKNIK 'AISYIYAH PONTIANAK

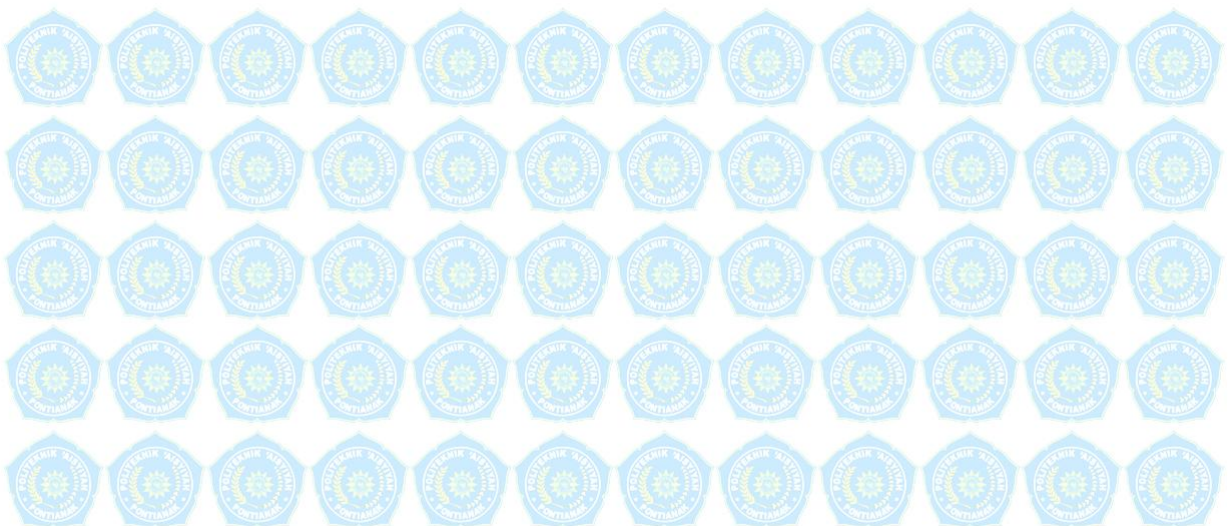
DAFTAR ISI

HALAMAN JUDUL	i
LEMBAR PERSETUJUAN	ii
LEMBAR PENGESAHAN.....	iii
PERNYATAAN INTEGRITAS AKADEMIK.....	iv
KATA PENGANTAR.....	v
DAFTAR ISI.....	vi
DAFTAR GAMBAR.....	viii
DAFTAR TABEL.....	ix
INTISARI.....	x
ABSTRACT	xi
BAB I PENDAHULUAN.....	1
2.1 Latar Belakang.....	1
2.2 Rumusan Masalah.....	3
2.3 Tujuan Penelitian.....	3
2.4 Ruang Lingkup	4
BAB II TINJAUAN PUSTAKA.....	6
2.1 Perbandingan Pustaka.....	6
2.2 Landasan Teori.....	9
2.2.1 Tanaman Kratom.....	9
2.2.2 Citra Digital.....	11
2.2.3 <i>Image Processing</i>	12
2.2.4 <i>Machine Learning</i>	13
2.2.5 <i>Convolutional Neural Network (CNN)</i>	14
2.2.6 <i>Grad-CAM (Gradient-weighted Class Activation Mapping)</i>	16
2.2.7 <i>Data Augmentation</i>	17
2.2.8 Klasifikasi Citra.....	18
2.2.9 <i>Dataset</i>	19

2.2.10	<i>Overfitting dan Underfitting</i>	20
2.2.11	Python	21
BAB III METODOLOGI PENELITIAN.....		22
4.1	Metode Perancangan.....	22
4.2	<i>Dataset</i> Citra.....	24
4.3	Proses Augmentasi Citra.....	25
4.4	Diagram Alir.....	27
4.4.1	Diagram Alir Sistem.....	29
4.5	Diagram <i>Use Case</i>	30
4.6	Metode Pengujian	31
BAB IV HASIL PENELITIAN DAN PEMBAHASAN.....		34
5.1	Hasil Penelitian	34
5.1.1	Hasil Pengumpulan <i>Dataset</i>	34
5.1.2	Hasil <i>Preprocessing</i> Citra.....	36
5.1.3	Hasil Augmentasi Citra	36
5.1.4	Hasil Pembagian <i>Dataset</i>	38
5.1.5	Hasil Uji dan Klasifikasi <i>Machine Learning</i>	39
5.1.6	Hasil <i>Confusion Matrix</i>	44
5.1.7	Hasil Pengujian Kompatibilitas Sistem.....	45
5.1.8	Hasil Visualisasi <i>Grad-CAM</i>	46
5.2	Pembahasan Hasil Penelitian	48
BAB V PENUTUP		51
6.1	Kesimpulan	51
6.2	Saran	52
DAFTAR PUSTAKA		54
LAMPIRAN.....		57

PERPUSTAKAAN

NPP. 6171052A2000001



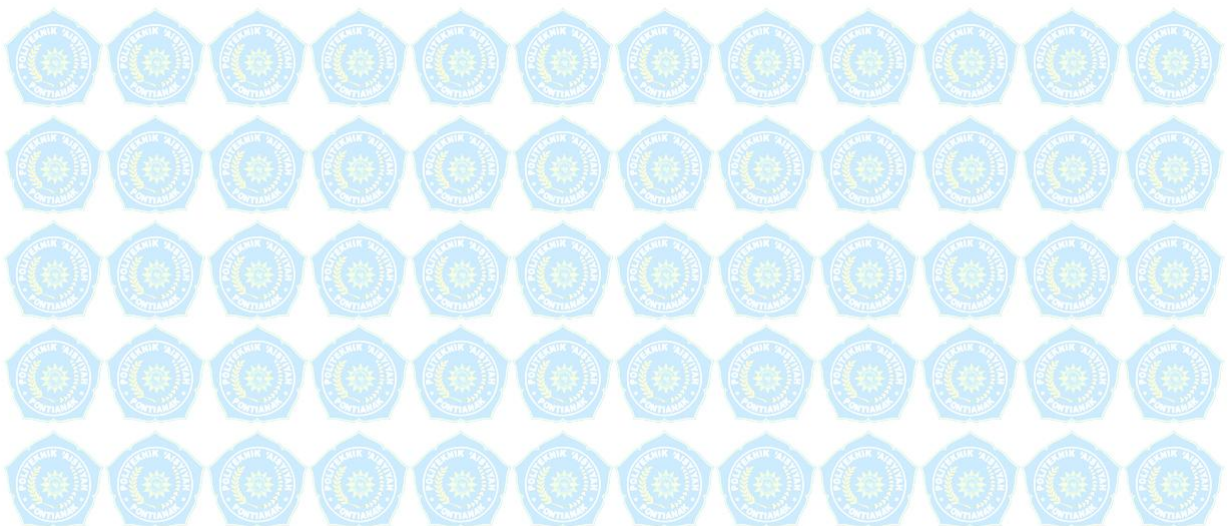
POLITEKNIK 'AISYIYAH PONTIANAK

DAFTAR GAMBAR

Gambar 2.1 Ciri-Ciri Daun Kratom	9
Gambar 2.2 Arsitektur CNN	14
Gambar 3.1 <i>Flowchart</i> Penelitian	28
Gambar 3.2 <i>Flowchart</i> Sistem	29
Gambar 3.3 Diagram <i>Use case</i>	31
Gambar 4.1 Dataset citra daun kratom dari <i>platform</i> Kaggle	34
Gambar 4.2 Dataset citra daun kratom hasil observasi	34
Gambar 4.3 Kode program preprocessing citra.....	35
Gambar 4.4 Kode program augmentasi citra.....	36
Gambar 4.5 ROC <i>Curve</i>	40
Gambar 4.6 Hasil <i>confusion matrix</i>	43
Gambar 4.7 Hasil <i>Grad-CAM</i> pada citra dengan kelas prediksi sehat.....	45
Gambar 4.8 Hasil <i>Grad-CAM</i> pada citra dengan kelas prediksi tidak sehat.....	46

PERPUSTAKAAN

NPP. 6171052A2000001



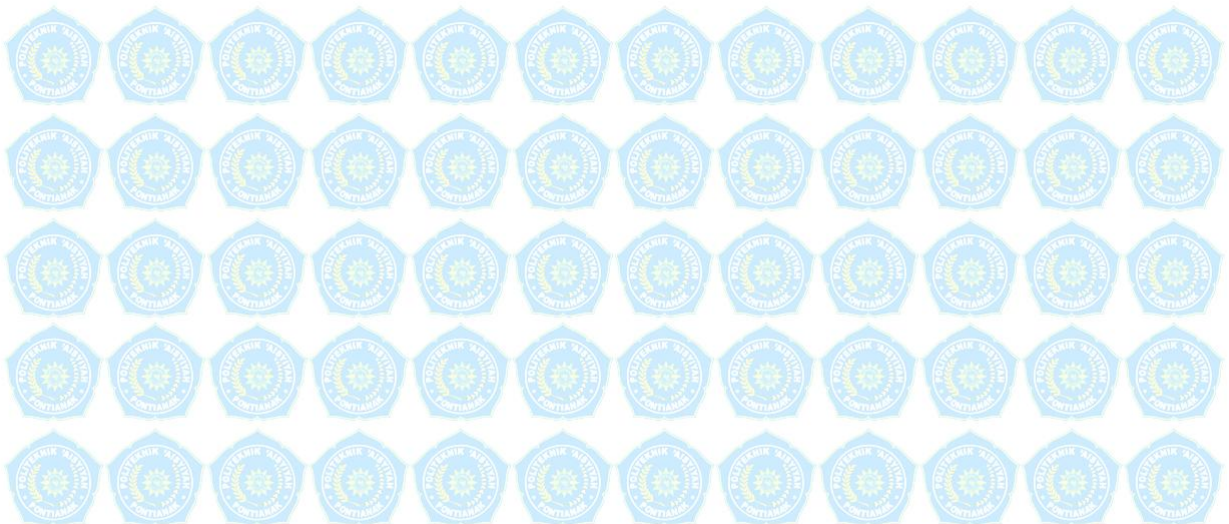
POLITEKNIK 'AISYIYAH PONTIANAK

DAFTAR TABEL

Tabel 2.1 Review Bandingan Pustaka	7
Tabel 3.1 Jumlah Citra Daun Kratom	25
Tabel 3.2 Rancangan Pengujian	32
Tabel 4.1 Pembagian Dataset	38
Tabel 4.2 Hasil Evaluasi Model <i>MobileNetV2</i> (5-Fold Cross Validation)	39
Tabel 4.3 <i>Classification Report</i>	41

PERPUSTAKAAN

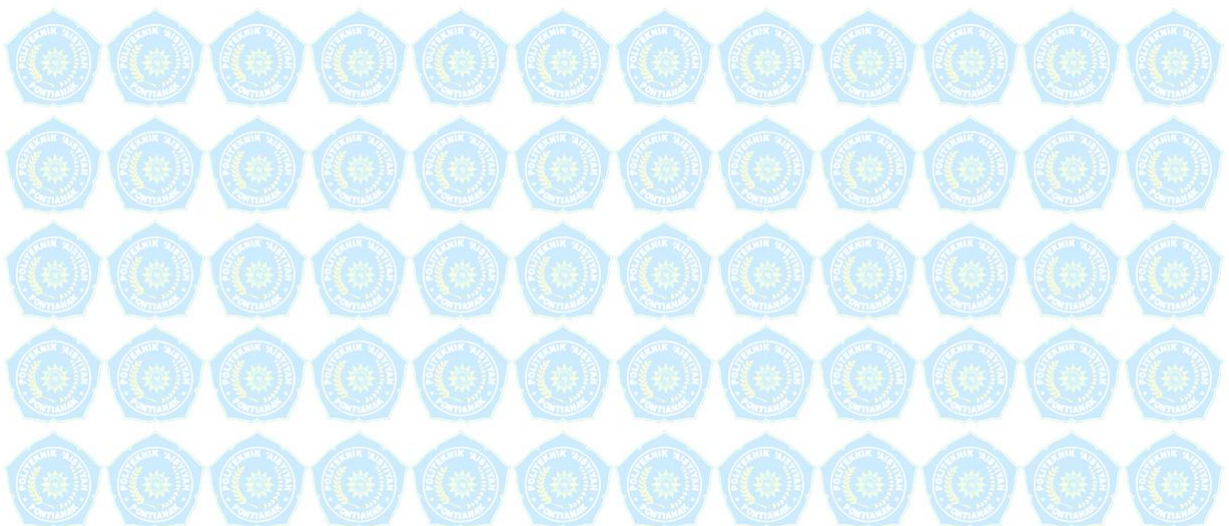
NPP. 6171052A2000001



POLITEKNIK 'AISYIYAH PONTIANAK

PERPUSTAKAAN

NPP. 6171052A2000001



POLITEKNIK 'AISYIYAH PONTIANAK

DAFTAR PUSTAKA

- Adawiyah, R., & Munifah. (2023). *Eksplorasi kapasitas pengkodean amplitudo untuk model quantum machine learning*. 3(1), 38–58. <http://journal.politeknik-pratama.ac.id/index.php/>
- Agustina, A., Yanto, F., Budianita, E., Iskandar, I., & Syafria, F. (2024). Klasifikasi penyakit tanaman padi menggunakan metode cnn arsitektur densenet-121 dan augmentasi data. *Journal Of Information Systems And Informatics Engineering*, 8(1). <https://doi.org/10.35145/joisie.v8i1.4256>
- Ayuni, D. P., Jasril, Irsyad, M., Yanto, F., & Sanjaya, S. (2023). *Augmentasi data pada implementasi convolutional neural network arsitektur EfficientNet-B3 untuk klasifikasi penyakit daun padi* (Vol. 5, Nomor 2).
- Badri, R. M., & Aningsih, S. (2025). *Buku ajar teknologi pengolahan citra digital*.
- Burhanuddin, I. (2024). *Pengembangan Media Pembelajaran Interaktif Berbasis Augmented Reality Untuk Meningkatkan Pemahaman Flowchart*. <https://jurnaldidaktika.org>
- Chandra, A. (2023). *Pengembangan aplikasi berbasis web dengan Python Flask untuk klasifikasi data halaman judu*.
- Dijaya, R. (2023). *Buku Ajar Pengolahan Citra Digital*.
- Fadllullah, A., & Natalia, T. (2026). Sistem Klasifikasi Citra Daun Obat Tradisional Dayak Kenyah Menggunakan Metode Convolutional Neural Network Dengan Arsitektur MobileNetV2. *Infomatek*, 28(1), 121–128. <https://doi.org/10.23969/infomatek.v28i1.36796>
- Fitria, D., Wardani, P. A., Uli, A. R. N., & Kurniawati, F. (2022). *Identifikasi dan Karakterisasi Tanaman Kratom melalui Pendekatan Profil Kandungan Senyawa Penanda secara LC-HRMS QToF dan Penetapan Nilai Retention Index secara GCMS*. 3(1), 79–90. <https://doi.org/10.54384/eru>
- Hakim, L., Rizaldi Rahmanto, H., Purnama Kristanto, S., & Yusuf, D. (2023). *Klasifikasi citra motif batik Banyuwangi menggunakan convolutional neural network* (Vol. 17, Nomor 1). <https://ejurnal.teknokrat.ac.id/index.php/teknoinfo/index>
- Hendry, Govert Anwar, F., Chow, D., Saputra, A., & Khaerul Naim Mursalim, M. (2025). Arsitektur Hybrid Vision Transformer-ConvNeXt dengan Multi-Task Focal Loss dan Medical Test-Time Augmentation untuk Klasifikasi Lesi Kulit Berbasis Citra. *Journal of Digital Ecosystem for Natural Sustainability*, 5(2), 55–66. <https://doi.org/10.63643/jodens.v5i2.325>
- Herawatiningsih, R., Sabtiani, R., & Eva, T. G. (2024). *Etnobotani tanaman kratom (Mitragyna speciosa) oleh masyarakat di sekitar DAS Labian di Desa Labian Ira'ang Kecamatan Batang Lupar Kabupaten Kapuas Hulu* (Vol. 12, Nomor 1).
- Hupont, I., Fernández-Llorca, D., Baldassarri, S., & Gómez, E. (2024). Use case cards: a use case reporting framework inspired by the European AI Act. *Ethics and Information Technology*, 26(2). <https://doi.org/10.1007/s10676-024-09757-7>
- Husen, D. (2024). *Evaluasi teknik augmentasi data untuk klasifikasi tumor otak menggunakan CNN pada citra MRI (Performance evaluation of CNN models*

with various data augmentation techniques on MRI images for brain tumor classification).

Ilham Suherman, M., Febrianti, R., Nurhidayah, N., Husnul Khatimah, F., & Ramdhany Edy, M. (2024). Analisis Sentimen Ekspresi Wajah : Pendekatan Deep Learning dan OpenCV dengan Python. *INTEC Journal: Information Technology Education Journal*, 3(3).

Jumadi, J., Yupianti, & Sartika, D. (2021). *Pengolahan citra digital untuk identifikasi objek menggunakan metode hierarchical agglomerative clustering*.

Kamilia Zulfiani, H., & Abdullah, A. (2025). Identifikasi Daun Herbal Untuk Tanaman Obat Menggunakan Metode Convolutional Neural Network (Cnn). Dalam *Jurnal Mahasiswa Teknik Informatika* (Vol. 9, Nomor 6).

Karim, M. J., Goni, M. O. F., Nahiduzzaman, M., Ahsan, M., Haider, J., & Kowalski, M. (2024). Enhancing agriculture through real-time grape leaf disease classification via an edge device with a lightweight CNN architecture and Grad-CAM. *Scientific Reports*, 14(1). <https://doi.org/10.1038/s41598-024-66989-9>

Khodijah, S., & Rizki, C. A. (2025). Analisis Kinerja Algoritma Deep Learning pada Pengolahan Data Kompleks. *Journal of Electrical Engineering Research*, 1(3). <https://doi.org/10.64803/joeer.v1i3.20>

López, M. A. O., López, A. M., & Crossa, J. (2022). Multivariate Statistical Machine Learning Methods for Genomic Prediction. Dalam *Multivariate Statistical Machine Learning Methods for Genomic Prediction*. Springer International Publishing. <https://doi.org/10.1007/978-3-030-89010-0>

Mursalin, R. A. (2025). *Segmentasi Semantik Menggunakan Model DeepLabv3+ untuk Citra Satelit pada Area Tutupan Lahan dan Objek Gunung Api*.

Muttaqien, M. Z. (2024). *Pandangan BNN terhadap penggunaan tanaman kratom pada komunitas Dayak di Kalimantan Barat*.

Normaidah, Najahidin, M., Rahmah, M., Fadlilaturrahmah, & Izma, H. (2023). Uji Sun Protection Factor (SPF) Ekstrak Etanol Daun *Mitragyna speciosa* Korth. *Jurnal Pharmascience*, 10(2), 386–393. <https://ppjp.ulm.ac.id/journal/index.php/pharmascience>

Nugraha, S. N., Pebrianto, R., & Fitri, E. (2023). Penerapan Deep Learning Pada Klasifikasi Tanaman Paprika Berdasarkan Citra Daun Menggunakan Metode CNN. *INFORMATION SYSTEM FOR EDUCATORS AND PROFESSIONALS*, 8(2), 8000063.

Oktafiani, R., Hermawan, A., & Avianto, D. (2023). Pengaruh Komposisi Split data Terhadap Performa Klasifikasi Penyakit Kanker Payudara Menggunakan Algoritma Machine Learning. *Jurnal Sains dan Informatika*, 19–28. <https://doi.org/10.34128/jsi.v9i1.622>

Pratama, W. A. (2023). *Regulasi terhadap penggunaan kratom (Mitragyna speciosa) sebagai narkotika jenis baru*.

Pratiwi, R. A. (2025). *Pendeteksian penyakit diabetic retinopathy melalui citra fundus menggunakan model CNN InceptionResNet-V2*.

Prevete, E., Kuypers, K. P. C., Theunissen, E. L., Esposito, G., Ramaekers, J. G., Pasquini, M., & Corazza, O. (2023). Clinical Implications of Kratom (*Mitragyna speciosa*) Use: a Literature Review. Dalam *Current Addiction*

- Reports* (Vol. 10, Nomor 2, hlm. 317–334). Springer Science and Business Media Deutschland GmbH. <https://doi.org/10.1007/s40429-023-00478-3>
- Putri, M. H., Zaky, U., & Prabawa, B. A. (2025). Optimizing Data Augmentation Parameters in YOLOv11 for Enhanced Rip Current Detection on Small Datasets from Depok-Parangtritis Coastline. *Jurnal Teknik Informatika (JUTIF)*, 6(5). <https://doi.org/10.52436/1.jutif.6.5.5352>
- Rabbani, F. F., Danang, M., Danu Arta, I., Rofiq, A., & Mukhsin, R. (2024). E-Commerce dan Legalitas Kratom: Implikasi Hukum bagi Pelaku Usaha di Pasar Domestik dan Internasional. Dalam *Indonesian Journal of Social Sciences and Humanities* (Vol. 4, Nomor 2). <https://bnn.go.id/bnn-e-commerce-marketplace-indonesia-satuan-persepsi/>
- Ramadhan, M. E., & Zeniarja, J. (2025). *Sistemasi: Jurnal Sistem Informasi Implementasi Deep Transfer Learning dan Explainable AI dalam Klasifikasi Kanker Kulit Implementation of Deep Transfer Learning and Explainable AI in Skin Cancer Classification*. <http://sistemasi.ftik.unisi.ac.id>
- Ramadhanti, Z. (2024). *Deteksi dan identifikasi jenis luka luar berdasarkan image feature menggunakan CNN dengan variasi pre-trained mode*.
- Roslan, N. A. M., Diah, N. M., Ibrahim, Z., Munarko, Y., & Minarno, A. E. (2023). Automatic plant recognition using convolutional neural network on Malaysian medicinal herbs: the value of data augmentation. *International Journal of Advances in Intelligent Informatics*, 9(1), 136–147. <https://doi.org/10.26555/ijain.v9i1.1076>
- Santoso, H., & Rochadiani, T. H. (2022). Pelatihan Machine Learning Menggunakan Bahasa Pemrograman Python Bagi Karyawan PT. Yokogawa Indonesia. *Jurnal ABDINUS: Jurnal Pengabdian Nusantara*, 6(2), 349–356. <https://doi.org/10.29407/ja.v6i2.16018>
- Sheila, S., Kharil Anwar, M., Saputra, A. B., Pujiyanto, R., & Sari, I. P. (2023). *Deteksi Penyakit pada Daun Padi Berbasis Pengolahan Citra Menggunakan Metode Convolutional Neural Network (CNN)*.
- Simon, G. (2024). *Health Informatics Artificial Intelligence and Machine Learning in Health Care and Medical Sciences Best Practices and Pitfalls*.
- Supriadi, I., & Mashudi, J. H. (2022). Analisis ekonomi keluarga petani kratom (*Mitragyna speciosa*) Desa Nanga Embaloh Kecamatan Embaloh Hilir Kabupaten Kapuas Hulu. *12*, 180–192. <https://doi.org/10.26418/jppk.v12i1.61721>
- Susanto, D., Natalia, D., & Widiyantoro, A. (2023). *Aktivitas antijamur ekstrak etanol daun kratom (Mitragyna speciosa) terhadap jamur Microsporum canis*.
- Syahrani, M. A., Budianto, T. A. C., & Adam, R. I. (2024). Klasifikasi buah segar dan busuk menggunakan algoritma convolutional neural network (CNN). Dalam *Jurnal Mahasiswa Teknik Informatika* (Vol. 8, Nomor 5).
- Syarma, R., Kartikawati, S. M., & Setyawati, D. (2023). *Karakteristik dan pengetahuan masyarakat Desa Entibab tentang pemanfaatan tumbuhan kratom (Mitragyna speciosa) di Kabupaten Kapuas Hulu*.
- Zarandah, Q. M. M., Mohd Daud, S., & Abu-Naser, S. S. (2023). Spectrogram flipping: A new technique for audio augmentation. *Journal of Theoretical and Applied Information Technology*, 15(11). www.jatit.org

LAMPIRAN

1. Kode Program *train_kfold.py*

```
1. """
2. =====
3. KLASIFIKASI DAUN KRATOM DENGAN 5-FOLD CROSS-VALIDATION
4. =====
5. Script ini mengatasi keterbatasan jumlah dataset asli (saat ini 100 citra:
6. 50 Healthy + 50 Unhealthy) dengan menerapkan Stratified 5-Fold Cross-
7. Validation, sehingga SETIAP citra asli mendapat giliran menjadi data
8. validasi tepat satu kali, dan confusion matrix akhir merupakan gabungan
9. hasil prediksi "out-of-fold" dari SELURUH citra asli (bukan cuma subset
10. 20% seperti pada split tunggal). Jumlah citra dibaca otomatis dari isi
11. folder dataset, jadi script ini tetap berjalan benar berapa pun jumlah
12. citra per kelas (tidak ada angka yang di-hardcode di kode).
13.
14. Alur untuk setiap fold:
15.     1. Seluruh citra asli dibagi menjadi ~80% train dan ~20% validasi
16.        (stratified, proporsi kelas dijaga tetap seimbang di setiap fold).
17.     2. Augmentasi dilakukan SECARA ONLINE (on-the-fly) menggunakan
18.        ImageDataGenerator langsung dari path citra asli -- TIDAK ADA lagi
19.        penyalinan/penulisan citra hasil augmentasi ke disk (tidak ada folder
20.        sementara seperti 'kfold_tmp'). Setiap batch training dibentuk dari
21.        transformasi acak (rotasi, flip, shift, zoom) atas citra asli.
22.     3. Data validasi fold ini TETAP citra asli, dibaca langsung tanpa
23.        augmentasi (hanya preprocessing MobileNetV2).
24.     4. Model MobileNetV2 dilatih dari awal (fresh) untuk fold ini (2 tahap:
25.        feature extraction + fine-tuning).
26.     5. Model memprediksi data validasi fold ini; hasilnya disimpan sebagai
27.        prediksi "out-of-fold" untuk citra-citra tersebut.
28.
29. Setelah 5 fold selesai:
30.     - Seluruh prediksi out-of-fold digabung -> classification report,
31.       confusion matrix, ROC/AUC, dan threshold optimal dihitung dari
32.       GABUNGAN seluruh citra ini (bukan cuma subset).
33.     - Model FINAL untuk deployment dilatih ulang sekali lagi menggunakan
34.       SELURUH citra asli (dengan augmentasi online penuh), agar model
35.       yang dipakai untuk prediksi nyata memanfaatkan semua data yang ada.
36.
37. Cara pakai:
38.     python train_kfold.py
39.
40. Hasil tersimpan di folder 'hasil_output/':
41.     classification_report.txt      (gabungan 5 fold, seluruh citra)
42.     confusion_matrix.png           (gabungan 5 fold, seluruh citra)
43.     roc_curve.png                 (gabungan 5 fold)
44.     ringkasan_metrik.json
45.     model_final_daun_kratom.h5     (dilatih ulang pada 100% data)
46.     label_map.json
47. =====
48. """
49.
50. import os
51. import json
52. import numpy as np
53. import pandas as pd
54. import matplotlib.pyplot as plt
55. import seaborn as sns
56. import tensorflow as tf
57. from sklearn.model_selection import StratifiedKFold
58. from tensorflow.keras.applications import MobileNetV2
59. from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
60. from tensorflow.keras.preprocessing.image import ImageDataGenerator
61. from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout,
Input
```

```

62. from tensorflow.keras.models import Model
63. from tensorflow.keras.optimizers import Adam
64. from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
65. from sklearn.metrics import (
66.     classification_report,
67.     confusion_matrix,
68.     roc_curve,
69.     auc,
70. )
71.
72. # =====
73. # 1. KONFIGURASI
74. # =====
75. DATASET_DIR = "dataset"          # folder citra ASLI (Healthy / Unhealthy)
76. OUTPUT_DIR = "hasil_output"
77. IMG_SIZE = (224, 224)
78. BATCH_SIZE = 32                  # dataset sekarang 100 citra (50/50) -> tiap fold
train ~80 citra,
79.                                # batch 32 masih aman & lebih stabil dibanding
dataset lama (64 citra)
80. EPOCHS = 30
81. LEARNING_RATE = 1e-4
82. FINE_TUNE_LR = 1e-5
83. FINE_TUNE_AT = 100
84. N_SPLITS = 5
85. RANDOM_SEED = 42
86.
87. CLASS_NAMES = ["Healthy", "Unhealthy"]
88.
89. os.makedirs(OUTPUT_DIR, exist_ok=True)
90. tf.random.set_seed(RANDOM_SEED)
91. np.random.seed(RANDOM_SEED)
92.
93. # =====
94. # 2. UTILITAS DATASET
95. # =====
96. def load_file_list():
97.     """Mengambil daftar path file dan label dari folder dataset asli."""
98.     filepaths, labels = [], []
99.     for idx, cls in enumerate(CLASS_NAMES):
100.         folder = os.path.join(DATASET_DIR, cls)
101.         for fname in sorted(os.listdir(folder)):
102.             if fname.lower().endswith((".jpg", ".jpeg", ".png", ".bmp")):
103.                 filepaths.append(os.path.join(folder, fname))
104.                 labels.append(idx)
105.     return np.array(filepaths), np.array(labels)
106.
107. def make_dataframe(filepaths, labels):
108.     """Membentuk DataFrame (filename, class) untuk dipakai
flow_from_dataframe."""
109.     return pd.DataFrame({
110.         "filename": filepaths,
111.         "class": [CLASS_NAMES[l] for l in labels],
112.     })
113.
114. def build_augmentation_datagen(validation_split=None):
115.     """ImageDataGenerator untuk augmentasi ONLINE (on-the-fly, tanpa tulis ke
disk)."""
116.     kwargs = dict(
117.         preprocessing_function=preprocess_input,
118.         rotation_range=90,
119.         width_shift_range=0.1,
120.         height_shift_range=0.1,
121.         shear_range=0.1,
122.         zoom_range=0.15,
123.         horizontal_flip=True,
124.         vertical_flip=True,

```

NPP. 6171052A2000001

```

125.         fill_mode="nearest",
126.     )
127.     if validation_split is not None:
128.         kwargs["validation_split"] = validation_split
129.     return ImageDataGenerator(**kwargs)
130.
131. def build_plain_datagen():
132.     """ImageDataGenerator TANPA augmentasi (dipakai untuk data validasi)."""
133.     return ImageDataGenerator(preprocessing_function=preprocess_input)
134.
135. # =====
136. # 3. MODEL
137. # =====
138. def build_model():
139.     base_model = MobileNetV2(input_shape=IMG_SIZE + (3,), include_top=False,
weights="imagenet")
140.     base_model.trainable = False
141.
142.     inputs = Input(shape=IMG_SIZE + (3,))
143.     x = base_model(inputs, training=False)
144.     x = GlobalAveragePooling2D()(x)
145.     x = Dropout(0.3)(x)
146.     x = Dense(128, activation="relu")(x)
147.     x = Dropout(0.2)(x)
148.     outputs = Dense(1, activation="sigmoid")(x)
149.
150.     model = Model(inputs, outputs)
151.     model.compile(optimizer=Adam(learning_rate=LEARNING_RATE),
152.                   loss="binary_crossentropy", metrics=["accuracy"])
153.     return model, base_model
154.
155. def train_two_stage(model, base_model, train_gen, val_gen):
156.     callbacks = [
157.         EarlyStopping(monitor="val_loss", patience=6,
restore_best_weights=True),
158.         ReduceLROnPlateau(monitor="val_loss", factor=0.5, patience=3,
min_lr=1e-7),
159.     ]
160.
161.     model.fit(train_gen, validation_data=val_gen, epochs=EPOCHS,
162.               callbacks=callbacks, verbose=2)
163.
164.     base_model.trainable = True
165.     for layer in base_model.layers[:FINE_TUNE_AT]:
166.         layer.trainable = False
167.     model.compile(optimizer=Adam(learning_rate=FINE_TUNE_LR),
168.                   loss="binary_crossentropy", metrics=["accuracy"])
169.     model.fit(train_gen, validation_data=val_gen, epochs=EPOCHS,
170.               callbacks=callbacks, verbose=2)
171.
172.     return model
173.
174. # =====
175. # 4. MAIN: 5-FOLD CROSS VALIDATION
176. # =====
177. def main():
178.     print("=" * 70)
179.     print("KLASIFIKASI DAUN KRATOM - 5-FOLD CROSS VALIDATION (AUGMENTASI
ONLINE)")
180.     print("=" * 70)
181.
182.     if not os.path.isdir(DATASET_DIR):
183.         raise FileNotFoundError(f"Folder dataset '{DATASET_DIR}' tidak
ditemukan.")
184.
185.     filepaths, labels = load_file_list()
186.     n_healthy = int(sum(labels == 0))

```

NPP. 6171052A2000001

```

187.     n_unhealthy = int(sum(labels == 1))
188.     print(f"\nTotal citra asli: {len(filepaths)} "
189.           f"({n_healthy} {CLASS_NAMES[0]}, {n_unhealthy} {CLASS_NAMES[1]})")
190.
191.     if n_healthy == 0 or n_unhealthy == 0:
192.         raise ValueError("Salah satu kelas kosong -- periksa isi folder
dataset/Healthy "
193.                           "dan dataset/Unhealthy.")
194.     if min(n_healthy, n_unhealthy) < N_SPLITS:
195.         raise ValueError(f"Jumlah citra kelas minoritas ({min(n_healthy,
n_unhealthy)}) "
196.                           f"lebih kecil dari N_SPLITS ({N_SPLITS}); Stratified
{N_SPLITS}-Fold "
197.                           "tidak bisa dibentuk.")
198.
199.     skf = StratifiedKFold(n_splits=N_SPLITS, shuffle=True,
random_state=RANDOM_SEED)
200.
201.     oof_true = np.zeros(len(filepaths), dtype=int)
202.     oof_prob = np.zeros(len(filepaths), dtype=float)
203.
204.     for fold_idx, (train_idx, val_idx) in enumerate(skf.split(filepaths,
labels), start=1):
205.         print(f"\n{'='*70}\nFOLD {fold_idx}/{N_SPLITS}\n{'='*70}")
206.
207.         train_df = make_dataframe(filepaths[train_idx], labels[train_idx])
208.         val_df = make_dataframe(filepaths[val_idx], labels[val_idx])
209.
210.         train_datagen = build_augmentation_datagen()
211.         val_datagen = build_plain_datagen()
212.
213.         train_gen = train_datagen.flow_from_dataframe(
214.             train_df, x_col="filename", y_col="class",
215.             target_size=IMG_SIZE, batch_size=BATCH_SIZE,
216.             class_mode="binary", classes=CLASS_NAMES,
217.             shuffle=True, seed=RANDOM_SEED,
218.         )
219.         # shuffle=False -> urutan val_gen SAMA dengan urutan val_df / val_idx,
220.         # sehingga hasil prediksi bisa langsung dipetakan balik ke
oof_true/oof_prob
221.         val_gen = val_datagen.flow_from_dataframe(
222.             val_df, x_col="filename", y_col="class",
223.             target_size=IMG_SIZE, batch_size=BATCH_SIZE,
224.             class_mode="binary", classes=CLASS_NAMES,
225.             shuffle=False,
226.         )
227.
228.         model, base_model = build_model()
229.         model = train_two_stage(model, base_model, train_gen, val_gen)
230.
231.         val_gen.reset()
232.         y_pred_prob_fold = model.predict(val_gen, verbose=0).ravel()
233.
234.         oof_true[val_idx] = labels[val_idx]
235.         oof_prob[val_idx] = y_pred_prob_fold
236.
237.         print(f"[INFO] Fold {fold_idx} selesai. Jumlah data validasi fold ini:
{len(val_idx)}")
238.
239.         tf.keras.backend.clear_session()
240.
241.         # =====
242.         # 5. EVALUASI GABUNGAN (OUT-OF-FOLD) - SELURUH CITRA
243.         # =====
244.         print(f"\n{'='*70}\nEVALUASI GABUNGAN (OUT-OF-FOLD, {len(filepaths)}
CITRA)\n{'='*70}")
245.

```

```

246.     fpr_tmp, tpr_tmp, thresholds_tmp = roc_curve(oof_true, oof_prob)
247.     youden_j = tpr_tmp - fpr_tmp
248.     best_threshold = float(thresholds_tmp[np.argmax(youden_j)])
249.     print(f"[INFO] Threshold optimal: {best_threshold:.4f}")
250.
251.     y_pred = (oof_prob >= best_threshold).astype(int)
252.
253.     report = classification_report(oof_true, y_pred, target_names=CLASS_NAMES,
digits=4)
254.     print("\n=== Classification Report (gabungan 5-fold) ===")
255.     print(report)
256.     with open(os.path.join(OUTPUT_DIR, "classification_report.txt"), "w") as f:
257.         f.write(report)
258.
259.     cm = confusion_matrix(oof_true, y_pred)
260.     plt.figure(figsize=(6, 5))
261.     sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
262.                 xticklabels=CLASS_NAMES, yticklabels=CLASS_NAMES)
263.     plt.title(f"Confusion Matrix - 5-Fold Cross Validation ({len(filepaths)}
citra)")
264.     plt.xlabel("Prediksi")
265.     plt.ylabel("Aktual")
266.     plt.tight_layout()
267.     plt.savefig(os.path.join(OUTPUT_DIR, "confusion_matrix.png"), dpi=200)
268.     plt.close()
269.
270.     fpr, tpr, _ = roc_curve(oof_true, oof_prob)
271.     roc_auc = auc(fpr, tpr)
272.     plt.figure(figsize=(6, 5))
273.     plt.plot(fpr, tpr, label=f"ROC curve (AUC = {roc_auc:.4f})", linewidth=2)
274.     plt.plot([0, 1], [0, 1], linestyle="--", color="gray", label="Random
Guess")
275.     plt.xlabel("False Positive Rate")
276.     plt.ylabel("True Positive Rate")
277.     plt.title("ROC Curve - 5-Fold Cross Validation")
278.     plt.legend(loc="lower right")
279.     plt.grid(True, alpha=0.3)
280.     plt.tight_layout()
281.     plt.savefig(os.path.join(OUTPUT_DIR, "roc_curve.png"), dpi=200)
282.     plt.close()
283.
284.     accuracy = float((oof_true == y_pred).mean())
285.     final_metrics = {
286.         "accuracy": accuracy,
287.         "auc": float(roc_auc),
288.         "best_threshold": best_threshold,
289.         "confusion_matrix": cm.tolist(),
290.         "class_names": CLASS_NAMES,
291.         "n_splits": N_SPLITS,
292.         "total_citra_asli": len(filepaths),
293.     }
294.     with open(os.path.join(OUTPUT_DIR, "ringkasan_metrik.json"), "w") as f:
295.         json.dump(final_metrics, f, indent=4)
296.
297.     print(f"\n[INFO] Accuracy (gabungan {len(filepaths)} citra):
{accuracy*100:.2f}%")
298.     print(f"[INFO] AUC: {roc_auc:.4f}")
299.
300.     # =====
301.     # 6. LATIH MODEL FINAL PADA 100% DATA (untuk deployment/prediksi nyata)
302.     # =====
303.     print(f"\n{' '*70}\nMELATIH MODEL FINAL PADA SELURUH {len(filepaths)} CITRA
ASLI\n{' '*70}")
304.
305.     full_df = make_dataframe(filepaths, labels)
306.
307.     # sisihkan sedikit data (10%) sekadar untuk monitoring EarlyStopping,

```

```

308.     # bukan untuk pelaporan metrik (metrik resmi = hasil 5-fold di atas)
309.     final_datagen = build_augmentation_datagen(validation_split=0.1)
310.
311.     final_train_gen = final_datagen.flow_from_dataframe(
312.         full_df, x_col="filename", y_col="class",
313.         target_size=IMG_SIZE, batch_size=BATCH_SIZE,
314.         class_mode="binary", classes=CLASS_NAMES,
315.         subset="training", shuffle=True, seed=RANDOM_SEED,
316.     )
317.     final_monitor_gen = final_datagen.flow_from_dataframe(
318.         full_df, x_col="filename", y_col="class",
319.         target_size=IMG_SIZE, batch_size=BATCH_SIZE,
320.         class_mode="binary", classes=CLASS_NAMES,
321.         subset="validation", shuffle=False, seed=RANDOM_SEED,
322.     )
323.
324.     final_model, final_base = build_model()
325.     final_model = train_two_stage(final_model, final_base, final_train_gen,
final_monitor_gen)
326.
327.     final_model_path = os.path.join(OUTPUT_DIR, "model_final_daun_kratom.h5")
328.     final_model.save(final_model_path)
329.     print(f"[INFO] Model final disimpan di: {final_model_path}")
330.
331.     with open(os.path.join(OUTPUT_DIR, "label_map.json"), "w") as f:
332.         json.dump({"0": CLASS_NAMES[0], "1": CLASS_NAMES[1]}, f, indent=4)
333.
334.     print("\n" + "=" * 70)
335.     print("SELESAI")
336.     print(f"Akurasi (5-fold, {len(filepaths)} citra) : {accuracy*100:.2f}%")
337.     print(f"AUC : {roc_auc:.4f}")
338.     print(f"Threshold optimal : {best_threshold:.4f}")
339.     print(f"Model final (dilatih pada 100% data) : {final_model_path}")
340.     print(f"Semua hasil tersimpan di folder: {OUTPUT_DIR}/")
341.     print("=" * 70)
342.
343. if __name__ == "__main__":
344.     main()

```

2. Kode Program *predict_visual.py*

```

1. """
2. =====
3. PREDIKSI + VISUALISASI - KLASIFIKASI PENYAKIT DAUN KRATOM
4. (Sehat / Tidak Sehat)
5. =====
6.
7. Script ini melakukan prediksi pada satu gambar maupun seluruh gambar dalam
8. suatu folder menggunakan model CNN yang telah dilatih.
9.
10. Fitur:
11. 1. Prediksi satu gambar atau seluruh gambar dalam folder.
12. 2. Menampilkan label hasil klasifikasi (Sehat / Tidak Sehat).
13. 3. Menampilkan persentase keyakinan model.
14. 4. Menyimpan gambar hasil prediksi.
15. 5. Membuat kolase seluruh hasil prediksi.
16. 6. Membuat grafik jumlah daun Sehat dan Tidak Sehat.
17. 7. Menyimpan tabel hasil prediksi dalam format CSV.
18.
19. Cara menjalankan:
20.
21. Prediksi satu gambar
22. python predict_visual.py --image gambar_uji/uji1.jpg

```

```

23.
24. Prediksi satu folder
25. python predict_visual.py --folder gambar_uji
26.
27. Semua hasil akan disimpan pada folder:
28. hasil_prediksi/
29. =====
30. """
31.
32. import os
33. import csv
34. import json
35. import argparse
36. import numpy as np
37. import matplotlib.pyplot as plt
38.
39. from tensorflow.keras.models import load_model
40. from tensorflow.keras.preprocessing import image
41. from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
42.
43. # =====
44. # KONFIGURASI
45. # =====
46.
47. MODEL_PATH = os.path.join("hasil_output", "model_final_daun_kratom.h5")
48. LABEL_MAP_PATH = os.path.join("hasil_output", "label_map.json")
49.
50. OUTPUT_DIR = "hasil_prediksi"
51. IMG_SIZE = (224, 224)
52.
53. os.makedirs(OUTPUT_DIR, exist_ok=True)
54.
55. _model_cache = None
56. _label_map_cache = None
57.
58. # =====
59. # MEMUAT MODEL
60. # =====
61.
62. def _load_resources():
63.     global _model_cache, _label_map_cache
64.
65.     if _model_cache is None:
66.         if not os.path.exists(MODEL_PATH):
67.             raise FileNotFoundError(
68.                 f"Model tidak ditemukan pada:\n{MODEL_PATH}\n\n"
69.                 "Silakan jalankan train_kfold.py terlebih dahulu."
70.             )
71.
72.         _model_cache = load_model(MODEL_PATH)
73.
74.     if _label_map_cache is None:
75.         with open(LABEL_MAP_PATH, "r") as f:
76.             _label_map_cache = json.load(f)
77.
78.     return _model_cache, _label_map_cache
79.
80. METRICS_PATH = os.path.join("hasil_output", "ringkasan_metrik.json")
81.
82. _threshold_cache = None
83.
84. def _load_threshold():
85.     """Baca threshold optimal (Youden's J) hasil 5-fold CV. Fallback ke 0.5
86.     jika tidak ada."""
87.     global _threshold_cache
88.     if _threshold_cache is None:
89.         if os.path.exists(METRICS_PATH):

```

```

89.         with open(METRICS_PATH, "r") as f:
90.             metrics = json.load(f)
91.             _threshold_cache = float(metrics.get("best_threshold", 0.5))
92.         else:
93.             print("[PERINGATAN] ringkasan_metrik.json tidak ditemukan, pakai
threshold default 0.5")
94.             _threshold_cache = 0.5
95.         return _threshold_cache
96.
97. # =====
98. # PREDIKSI SATU GAMBAR
99. # =====
100.
101. def predict_single(image_path):
102.
103.     model, label_map = _load_resources()
104.
105.     img = image.load_img(image_path, target_size=IMG_SIZE)
106.
107.     img_array = image.img_to_array(img)
108.     img_array = np.expand_dims(img_array, axis=0)
109.     img_array = preprocess_input(img_array)
110.
111.     probability = model.predict(img_array, verbose=0)[0][0]
112.
113.     threshold = _load_threshold()
114.     predicted_class = "1" if probability >= threshold else "0"
115.
116.     label_en = label_map[predicted_class]
117.
118.     confidence = (
119.         probability * 100
120.         if predicted_class == "1"
121.         else (1 - probability) * 100
122.     )
123.
124.     label_id = (
125.         "SEHAT"
126.         if label_en == "Healthy"
127.         else "TIDAK SEHAT"
128.     )
129.
130.     return label_id, float(confidence), img
131.
132. # =====
133. # SIMPAN HASIL SATU GAMBAR
134. # =====
135.
136. def save_single_result_image(filename, label, confidence, img):
137.
138.     warna = "green" if label == "SEHAT" else "red"
139.
140.     fig, ax = plt.subplots(figsize=(5, 6))
141.
142.     ax.imshow(img)
143.     ax.axis("off")
144.
145.     ax.set_title(
146.         f"Hasil: {label}\nKeyakinan: {confidence:.2f}%",
147.         fontsize=13,
148.         color=warna,
149.         fontweight="bold"
150.     )
151.
152.     fig.text(
153.         0.5,
154.         0.03,

```

NPP. 6171052A2000001

```

155.         f"Berdasarkan hasil klasifikasi model CNN, "
156.         f"daun kratom termasuk kategori {label} "
157.         f"dengan tingkat keyakinan {confidence:.2f}%",
158.         ha="center",
159.         fontsize=9,
160.         wrap=True
161.     )
162.
163.     plt.tight_layout(rect=[0, 0.06, 1, 1])
164.
165.     save_path = os.path.join(
166.         OUTPUT_DIR,
167.         os.path.splitext(os.path.basename(filename))[0] + "_hasil.png"
168.     )
169.
170.     plt.savefig(save_path, dpi=200)
171.     plt.close()
172.
173.     return save_path
174.
175. def save_grid_result(results):
176.
177.     if len(results) == 0:
178.         return None
179.
180.     cols = min(4, len(results))
181.     rows = int(np.ceil(len(results) / cols))
182.
183.     fig, axes = plt.subplots(
184.         rows,
185.         cols,
186.         figsize=(cols * 3.5, rows * 3.8)
187.     )
188.
189.     axes = np.array(axes).reshape(-1)
190.
191.     for ax, r in zip(axes, results):
192.
193.         warna = "green" if r["label"] == "SEHAT" else "red"
194.
195.         ax.imshow(r["img"])
196.         ax.axis("off")
197.
198.         ax.set_title(
199.             f"{r['file']}\n{r['label']} ({r['confidence']:.2f}%",
200.             fontsize=9,
201.             color=warna
202.         )
203.
204.     for ax in axes[len(results):]:
205.         ax.axis("off")
206.
207.     plt.tight_layout()
208.
209.     save_path = os.path.join(
210.         OUTPUT_DIR,
211.         "kolase_hasil_prediksi.png"
212.     )
213.
214.     plt.savefig(save_path, dpi=200)
215.     plt.close()
216.
217.     return save_path
218.
219. # =====
220. # GRAFIK RINGKASAN
221. # =====

```

NPP. 6171052A2000001

```

222.
223. def save_summary_chart(results):
224.
225.     sehat = sum(r["label"] == "SEHAT" for r in results)
226.     tidak = sum(r["label"] == "TIDAK SEHAT" for r in results)
227.
228.     fig, ax = plt.subplots(figsize=(5, 4))
229.
230.     bars = ax.bar(
231.         ["Sehat", "Tidak Sehat"],
232.         [sehat, tidak],
233.         color=["#1DB954", "#E63946"]
234.     )
235.
236.     for b in bars:
237.         ax.text(
238.             b.get_x() + b.get_width()/2,
239.             b.get_height() + 0.05,
240.             int(b.get_height()),
241.             ha="center",
242.             fontweight="bold"
243.         )
244.
245.     ax.set_title(
246.         "Ringkasan Hasil Prediksi Daun Kratom",
247.         fontsize=12,
248.         fontweight="bold"
249.     )
250.
251.     ax.set_ylabel("Jumlah Gambar")
252.
253.     plt.tight_layout()
254.
255.     save_path = os.path.join(
256.         OUTPUT_DIR,
257.         "ringkasan_prediksi.png"
258.     )
259.
260.     plt.savefig(save_path, dpi=200)
261.     plt.close()
262.
263.     return save_path, sehat, tidak
264.
265. # =====
266. # CSV
267. # =====
268.
269. def save_summary_csv(results):
270.
271.     csv_path = os.path.join(
272.         OUTPUT_DIR,
273.         "ringkasan_prediksi.csv"
274.     )
275.
276.     with open(csv_path, "w", newline="") as f:
277.
278.         writer = csv.writer(f)
279.
280.         writer.writerow([
281.             "Nama File",
282.             "Hasil Klasifikasi",
283.             "Keyakinan (%)"
284.         ])
285.
286.         for r in results:
287.
288.             writer.writerow([

```

NPP. 6171052A2000001

```

289.         r["file"],
290.         r["label"],
291.         f"{r['confidence']:.2f}"
292.     ])
293.
294.     return csv_path
295.
296. # =====
297. # PREDIKSI SATU FOLDER
298. # =====
299.
300. def run_on_folder(folder):
301.
302.     ekstensi = (".jpg", ".jpeg", ".png", ".bmp")
303.
304.     results = []
305.
306.     for file in sorted(os.listdir(folder)):
307.
308.         if not file.lower().endswith(ekstensi):
309.             continue
310.
311.         path = os.path.join(folder, file)
312.
313.         label, conf, img = predict_single(path)
314.
315.         save_single_result_image(file, label, conf, img)
316.
317.         print(f"{file:30s} -> {label:12s} ({conf:.2f}%)")
318.
319.         results.append({
320.             "file": file,
321.             "label": label,
322.             "confidence": conf,
323.             "img": img
324.         })
325.
326.     if len(results) == 0:
327.         print("Tidak ada gambar ditemukan.")
328.         return
329.
330.     grid = save_grid_result(results)
331.     chart, sehat, tidak = save_summary_chart(results)
332.     csv_file = save_summary_csv(results)
333.
334.     print("\n==== RINGKASAN =====")
335.     print(f"Total gambar : {len(results)}")
336.     print(f"Sehat : {sehat}")
337.     print(f"Tidak Sehat : {tidak}")
338.
339.     print("\nSemua hasil tersimpan pada folder:")
340.     print(OUTPUT_DIR)
341.     print(grid)
342.     print(chart)
343.     print(csv_file)
344.
345. # =====
346. # PREDIKSI SATU GAMBAR
347. # =====
348.
349. def run_on_single(image_path):
350.
351.     label, conf, img = predict_single(image_path)
352.
353.     hasil = save_single_result_image(
354.         image_path,
355.         label,

```

NPP. 6171052A2000001

```

356.         conf,
357.         img
358.     )
359.
360.     print("\n===== HASIL PREDIKSI =====")
361.     print(f"Gambar      : {image_path}")
362.     print(f"Hasil        : {label}")
363.     print(f"Keyakinan    : {conf:.2f}%")
364.     print(f"Hasil visual: {hasil}")
365.
366. # =====
367. # MAIN
368. # =====
369.
370. if __name__ == "__main__":
371.
372.     parser = argparse.ArgumentParser(
373.         description="Prediksi kesehatan daun kratom menggunakan model CNN"
374.     )
375.
376.     parser.add_argument(
377.         "--image",
378.         type=str,
379.         help="Path satu gambar daun kratom"
380.     )
381.
382.     parser.add_argument(
383.         "--folder",
384.         type=str,
385.         help="Path folder gambar daun kratom"
386.     )
387.
388.     args = parser.parse_args()
389.
390.     if args.image:
391.         run_on_single(args.image)
392.
393.     elif args.folder:
394.         run_on_folder(args.folder)
395.
396.     else:
397.         print(
398.             "Gunakan:\n"
399.             "python predict_visual.py --image gambar.jpg\n"
400.             "atau\n"
401.             "python predict_visual.py --folder gambar_uji"
402.         )

```

3. Kode Program *predict_explain.py*

```

1. """
2. =====
3. PREDIKSI + PENJELASAN VISUAL (VERSI PREMIUM) - DAUN KRATOM
4. =====
5. Peningkatan dari versi sebelumnya:
6.     - Layout kartu (card) bergaya dashboard modern, bukan sekadar 3 panel polos
7.     - Gauge/speedometer keyakinan model (visual, bukan cuma angka)
8.     - Progress bar tingkat keparahan bercak
9.     - Badge status berwarna (hijau/merah) dengan ikon check/warning
10.    - Anotasi kontur area bercak (bukan cuma overlay blok warna)
11.    - Watermark judul & tanggal analisis otomatis di setiap gambar output
12.    - Ringkasan folder dalam bentuk dashboard (bukan cuma bar chart biasa)
13.    - Menggunakan threshold optimal (Youden's J) hasil 5-fold CV, bukan 0.5 fix

```

```

14.     - Grad-CAM diperbaiki agar berfungsi pada base_model bersarang
(MobileNetV2)
15.
16. Cara pakai:
17.     python predict_explain.py --image gambar_uji/uji.jpg
18.     python predict_explain.py --folder gambar_uji
19.
20. Hasil tersimpan di folder: hasil_penjelasan/
21. =====
22. """
23.
24. import os
25. import json
26. import argparse
27. import datetime
28. import numpy as np
29. import matplotlib.pyplot as plt
30. import matplotlib.colors as mcolors
31. import matplotlib.patches as mpatches
32. from matplotlib.gridspec import GridSpec
33. from matplotlib.patches import Wedge, Circle
34. import tensorflow as tf
35. from tensorflow.keras.models import load_model
36. from tensorflow.keras.preprocessing import image
37. from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
38.
39. MODEL_PATH = os.path.join("hasil_output", "model_final_daun_kratom.h5")
40. LABEL_MAP_PATH = os.path.join("hasil_output", "label_map.json")
41. METRICS_PATH = os.path.join("hasil_output", "ringkasan_metrik.json")
42. OUTPUT_DIR = "hasil_penjelasan"
43. IMG_SIZE = (224, 224)
44. SPOT_THRESHOLD_PERSEN = 5.0
45.
46. # Ubah ke True kalau ingin melihat traceback lengkap ketika Grad-CAM gagal
dihitung
47. DEBUG_GRADCAM = False
48.
49. # --- Palet warna tema dashboard ---
50. WARNA_SEHAT = "#1DB954"
51. WARNA_TIDAK_SEHAT = "#E63946"
52. WARNA_BG_CARD = "#F5F7FA"
53. WARNA_TEKS = "#1B1B1F"
54. WARNA_AKSEN = "#264653"
55.
56. os.makedirs(OUTPUT_DIR, exist_ok=True)
57.
58. _model_cache = None
59. _label_map_cache = None
60. _threshold_cache = None
61. _last_conv_name_cache = None
62.
63. # =====
64. # 1. LOAD MODEL, LABEL & THRESHOLD
65. # =====
66. def _load_resources():
67.     global _model_cache, _label_map_cache
68.     if _model_cache is None:
69.         if not os.path.exists(MODEL_PATH):
70.             raise FileNotFoundError(
71.                 f"Model tidak ditemukan di '{MODEL_PATH}'. "
72.                 f"Jalankan train_kfold.py terlebih dahulu."
73.             )
74.         _model_cache = load_model(MODEL_PATH)
75.     if _label_map_cache is None:
76.         with open(LABEL_MAP_PATH, "r") as f:
77.             _label_map_cache = json.load(f)
78.     return _model_cache, _label_map_cache

```

```

79.
80. def _load_threshold():
81.     """Baca threshold optimal (Youden's J) hasil 5-fold CV. Fallback ke 0.5
jika tidak ada."""
82.     global _threshold_cache
83.     if _threshold_cache is None:
84.         if os.path.exists(METRICS_PATH):
85.             with open(METRICS_PATH, "r") as f:
86.                 metrics = json.load(f)
87.                 _threshold_cache = float(metrics.get("best_threshold", 0.5))
88.         else:
89.             print("[PERINGATAN] ringkasan_metrik.json tidak ditemukan, pakai
threshold default 0.5")
90.             _threshold_cache = 0.5
91.     return _threshold_cache
92.
93. def _find_nested_base_model(model):
94.     for layer in model.layers:
95.         if isinstance(layer, tf.keras.Model):
96.             return layer
97.     return None
98.
99. def _find_last_conv_layer_name(base_model):
100.     """
101.     Cari nama layer terakhir dengan output 4D (feature map konvolusi),
102.     dicari di dalam graf INTERNAL base_model sendiri.
103.     Aman dipakai di sini karena kita hanya butuh SHAPE-nya (bukan tensor
yang harus tersambung ke model luar). get_output_shape_at/get_output_at
sudah dihapus di Keras 3, jadi kita pakai layer.output.shape langsung
-- ini valid karena dalam graf internal base_model sendiri tiap layer
cuma dipanggil sekali (1 node), tidak ambigu.
104.     """
105.     for layer in reversed(base_model.layers):
106.         try:
107.             shape = layer.output.shape
108.         except Exception:
109.             continue
110.         if shape is not None and len(shape) == 4:
111.             return layer.name
112.     return None
113.
114. # =====
115. # 2. GRAD-CAM
116. # =====
117. _head_layers_cache = None
118.
119. def _find_head_layers(model, base_model):
120.     """
121.     Ambil layer-layer 'kepala' (setelah base_model) dari model utama,
122.     berurutan sesuai posisinya di model.layers -- misal:
123.     GlobalAveragePooling2D, Dropout, Dense, Dropout, Dense.
124.     """
125.     head_layers = []
126.     found_base = False
127.     for layer in model.layers:
128.         if layer is base_model:
129.             found_base = True
130.             continue
131.         if found_base:
132.             head_layers.append(layer)
133.     return head_layers
134.
135. def compute_gradcam(model, img_array_prep):
136.     """
137.     Grad-CAM untuk model dengan base_model (MobileNetV2) bersarang.
138.
139.     Di Keras 3, layer.output pada layer di DALAM base_model mengarah ke

```

```

144.     graf internal base_model sendiri, TIDAK tersambung ke input model luar
145.     -- jadi tidak bisa langsung dipakai membuat satu grad_model gabungan
146.     seperti Grad-CAM biasa (akan muncul error "Output ... is not connected
147.     to inputs").
148.
149.     Solusinya: hitung Grad-CAM dalam 2 tahap, di dalam GradientTape yang sama:
150.     1) Jalankan base_model (lewat sub-model barunya sendiri: base_model.input
151.         -> layer konvolusi terakhir) untuk dapat feature map konvolusi.
152.     2) Lanjutkan manual layer-layer 'kepala' (GAP, Dropout, Dense, dst.)
153.         satu per satu di atas feature map tadi untuk dapat prediksi akhir.
154.     Gradient prediksi terhadap feature map dari langkah 1 itulah Grad-CAM.
155.     """
156.     global _last_conv_name_cache, _head_layers_cache
157.     try:
158.         base_model = _find_nested_base_model(model)
159.         if base_model is None:
160.             return None
161.         if _last_conv_name_cache is None:
162.             _last_conv_name_cache = _find_last_conv_layer_name(base_model)
163.         if _last_conv_name_cache is None:
164.             return None
165.         if _head_layers_cache is None:
166.             _head_layers_cache = _find_head_layers(model, base_model)
167.         if not _head_layers_cache:
168.             return None
169.
170.         conv_extractor = tf.keras.models.Model(
171.             inputs=base_model.input,
172.             outputs=base_model.get_layer(_last_conv_name_cache).output,
173.         )
174.
175.         img_tensor = tf.convert_to_tensor(img_array_prep)
176.
177.         with tf.GradientTape() as tape:
178.             tape.watch(img_tensor)
179.             conv_output = conv_extractor(img_tensor)
180.             x = conv_output
181.             for layer in _head_layers_cache:
182.                 x = layer(x, training=False)
183.             predictions = x
184.             loss = predictions[:, 0]
185.
186.             grads = tape.gradient(loss, conv_output)
187.             if grads is None:
188.                 return None
189.
190.             pooled_grads = tf.reduce_mean(grads, axis=(0, 1, 2))
191.             conv_output_single = conv_output[0]
192.             heatmap = conv_output_single @ pooled_grads[..., tf.newaxis]
193.             heatmap = tf.squeeze(heatmap)
194.             heatmap = tf.maximum(heatmap, 0) / (tf.reduce_max(heatmap) + 1e-8)
195.             heatmap = tf.image.resize(heatmap[..., tf.newaxis], IMG_SIZE)
196.             return heatmap.numpy().squeeze()
197.     except Exception as e:
198.         if DEBUG_GRADCAM:
199.             import traceback
200.             traceback.print_exc()
201.         print(f"[PERINGATAN] Grad-CAM tidak dapat dihitung ({e}). Melanjutkan
202.             tanpa heatmap.")
203.         return None
204.
205. # =====
206. # 3. DETEKSI BERCAK WARNA
207. # =====
208. def detect_spots(img_array_uint8):
209.     rgb_norm = img_array_uint8.astype("float32") / 255.0
210.     hsv = mcolors.rgb_to_hsv(rgb_norm)

```

```

210.     h, s, v = hsv[..., 0], hsv[..., 1], hsv[..., 2]
211.     leaf_mask = s > 0.15
212.     spot_mask = leaf_mask & (h < 0.17) & (v > 0.15)
213.     total_leaf_pixels = int(leaf_mask.sum())
214.     total_spot_pixels = int(spot_mask.sum())
215.     if total_leaf_pixels == 0:
216.         return 0.0, spot_mask
217.     return (total_spot_pixels / total_leaf_pixels) * 100, spot_mask
218.
219. def get_severity_level(label):
220.     if label == "SEHAT":
221.         return "Sehat", "#1DB954"
222.     else:
223.         return "Tidak Sehat", "#E63946"
224.
225. # =====
226. # 4. PREDIKSI + KETERANGAN
227. # =====
228. def predict_and_explain(image_path):
229.     model, label_map = _load_resources()
230.     threshold = _load_threshold()
231.
232.     img_pil = image.load_img(image_path, target_size=IMG_SIZE)
233.     img_array = image.img_to_array(img_pil)
234.     img_array_batch = np.expand_dims(img_array, axis=0)
235.     img_array_prep = preprocess_input(img_array_batch.copy())
236.
237.     pred_prob = model.predict(img_array_prep, verbose=0)[0][0]
238.     predicted_class = "1" if pred_prob >= threshold else "0"
239.     label_en = label_map[predicted_class]
240.
241.     confidence = pred_prob * 100 if predicted_class == "1" else (1 - pred_prob)
242.     * 100
243.     label_id = "SEHAT" if label_en == "Healthy" else "TIDAK SEHAT"
244.
245.     heatmap = compute_gradcam(model, img_array_prep)
246.     persen_bercak, spot_mask = detect_spots(img_array)
247.
248.     if label_id == "SEHAT":
249.         ket_bercak = (
250.             f"Tidak ditemukan indikasi bercak yang signifikan "
251.             f"({persen_bercak:.1f})% area terindikasi)."
252.         )
253.     else:
254.         ket_bercak = (
255.             f"Ditemukan indikasi bercak pada sekitar "
256.             f"({persen_bercak:.1f})% permukaan daun."
257.         )
258.
259.     keterangan = (
260.         f"Berdasarkan hasil analisis citra menggunakan model CNN, "
261.         f"daun kratom diklasifikasikan sebagai {label_id} "
262.         f"dengan tingkat keyakinan {confidence:.2f}%."
263.         f"{ket_bercak}"
264.     )
265.
266.     return {
267.         "label": label_id,
268.         "confidence": confidence,
269.         "persen_bercak": persen_bercak,
270.         "keterangan": keterangan,
271.         "img": img_array.astype("uint8"),
272.         "heatmap": heatmap,
273.         "spot_mask": spot_mask,
274.     }
275. # =====

```

NPP. 6171052A2000001

```

276. # 5. KOMPONEN VISUAL - GAUGE KEYAKINAN (speedometer)
277. # =====
278. def draw_confidence_gauge(ax, confidence, color):
279.     ax.set_xlim(-1.2, 1.2)
280.     ax.set_ylim(-0.2, 1.2)
281.     ax.axis("off")
282.     ax.set_aspect("equal")
283.
284.     # track abu-abu penuh (0-100%)
285.     ax.add_patch(Wedge((0, 0), 1.0, 0, 180, width=0.28,
286.                        facecolor="#E0E0E0", edgecolor="none"))
287.     # progress sesuai confidence
288.     sudut = 180 * (confidence / 100)
289.     ax.add_patch(Wedge((0, 0), 1.0, 180 - sudut, 180,
290.                        width=0.28, facecolor=color, edgecolor="none"))
291.     # jarum
292.     ax.add_patch(Circle((0, 0), 0.05, facecolor=WARN_AKSEN, zorder=5))
293.
294.     ax.text(0, 0.42, f"{confidence:.1f}%", ha="center", va="center",
295.            fontsize=20, fontweight="bold", color=WARN_AKSEN)
296.     ax.text(0, 0.15, "Tingkat Keyakinan", ha="center", va="center",
297.            fontsize=9, color="#555555")
298.
299. def draw_severity_bar(ax, persen_bercak):
300.     ax.set_xlim(0, 100)
301.     ax.set_ylim(0, 1)
302.     ax.axis("off")
303.
304.     ax.add_patch(
305.         mpatches.FancyBboxPatch(
306.             (0, 0.25), 100, 0.5,
307.             boxstyle="round,pad=0,rounding_size=0.25",
308.             facecolor="#E0E0E0",
309.             edgecolor="none"
310.         )
311.     )
312.
313.     ax.add_patch(
314.         mpatches.FancyBboxPatch(
315.             (0, 0.25), min(persen_bercak, 100), 0.5,
316.             boxstyle="round,pad=0,rounding_size=0.25",
317.             facecolor="#E63946",
318.             edgecolor="none"
319.         )
320.     )
321.
322.     ax.text(
323.         0,
324.         1.0,
325.         f"Area Bercak Terdeteksi : {persen_bercak:.1f}%",
326.         fontsize=10,
327.         fontweight="bold",
328.         color=WARN_AKSEN,
329.         va="bottom"
330.     )
331.
332. def draw_status_badge(ax, label, confidence):
333.     ax.axis("off")
334.     ax.set_xlim(0, 1)
335.     ax.set_ylim(0, 1)
336.     color = WARN_SEHAT if label == "SEHAT" else WARN_TIDAK_SEHAT
337.     ikon = "\u2713" if label == "SEHAT" else "\u26A0"
338.     ax.add_patch(mpatches.FancyBboxPatch((0.05, 0.15), 0.9, 0.7,
339.                                           boxstyle="round,pad=0.02,rounding_size
340.                                           =0.15",
341.                                           facecolor=color, edgecolor="none",
342.                                           alpha=0.15))

```

NPP. 6171052A2000001

```

341.     ax.text(0.5, 0.5, f"{ikon} {label}", ha="center", va="center",
342.             fontsize=22, fontweight="bold", color=color)
343.
344. # =====
345. # 6. FIGUR UTAMA - DASHBOARD SATU GAMBAR
346. # =====
347. def save_explanation_figure(filename, result):
348.     color = WARNA_SEHAT if result["label"] == "SEHAT" else WARNA_TIDAK_SEHAT
349.     timestamp = datetime.datetime.now().strftime("%d %B %Y, %H:%M")
350.
351.     fig = plt.figure(figsize=(13, 8), facecolor="white")
352.     gs = GridSpec(3, 4, figure=fig, height_ratios=[0.55, 3, 0.9],
353.                  width_ratios=[1, 1, 1, 1.1], hspace=0.45, wspace=0.25)
354.
355.     # --- Header ---
356.     ax_header = fig.add_subplot(gs[0, :])
357.     ax_header.axis("off")
358.     ax_header.text(0.0, 0.7, "LAPORAN ANALISIS DAUN KRATOM", fontsize=17,
359.                  fontweight="bold", color=WARNA_AKSEN, ha="left")
360.     ax_header.text(0.0, 0.15, os.path.basename(filename), fontsize=10.5,
361.                  color="#555555", ha="left")
362.     ax_header.text(1.0, 0.7, timestamp, fontsize=9.5,
363.                  color="#888888", ha="right")
364.     ax_header.axhline(y=0, xmin=0, xmax=1, color="#DDDDDD", linewidth=1.2)
365.
366.     # --- Panel gambar: asli, Grad-CAM, deteksi bercak ---
367.     ax1 = fig.add_subplot(gs[1, 0])
368.     ax1.imshow(result["img"])
369.     ax1.axis("off")
370.     ax1.set_title("Citra Asli", fontsize=11,
371.                 fontweight="bold", color=WARNA_AKSEN)
372.
373.     ax2 = fig.add_subplot(gs[1, 1])
374.     if result["heatmap"] is not None:
375.         ax2.imshow(result["img"])
376.         ax2.imshow(result["heatmap"], cmap="jet", alpha=0.5)
377.         ax2.set_title("Grad-CAM\n(Fokus Model)", fontsize=11,
378.                     fontweight="bold", color=WARNA_AKSEN)
379.     else:
380.         ax2.imshow(result["img"])
381.         ax2.set_title("Grad-CAM\n(tidak tersedia)",
382.                     fontsize=11, color="#999999")
383.     ax2.axis("off")
384.
385.     ax3 = fig.add_subplot(gs[1, 2])
386.     overlay = result["img"].copy()
387.     overlay[result["spot_mask"]] = [255, 0, 200]
388.     ax3.imshow(overlay)
389.     ax3.axis("off")
390.     ax3.set_title("Deteksi Bercak\n(Analisis Warna)",
391.                 fontsize=11, fontweight="bold", color=WARNA_AKSEN)
392.
393.     # --- Panel kanan: badge status + gauge keyakinan ---
394.     ax_badge = fig.add_subplot(gs[1, 3])
395.     ax_badge.set_position([gs[1, 3].get_position(fig).x0,
396.                          gs[1, 3].get_position(
397.                              fig).y0 + gs[1, 3].get_position(fig).height *
398.                              0.62,
399.                          gs[1, 3].get_position(fig).width,
400.                          gs[1, 3].get_position(fig).height * 0.35])
401.     draw_status_badge(ax_badge, result["label"], result["confidence"])
402.
403.     ax_gauge = fig.add_axes([gs[1, 3].get_position(fig).x0,
404.                             gs[1, 3].get_position(fig).y0,
405.                             gs[1, 3].get_position(fig).width,
406.                             gs[1, 3].get_position(fig).height * 0.55])
407.     draw_confidence_gauge(ax_gauge, result["confidence"], color)

```

```

407.
408.     # --- Bar tingkat keparahan bercak (full width) ---
409.     ax_bar = fig.add_subplot(gs[2, :2])
410.     draw_severity_bar(ax_bar, result["persen_bercak"])
411.
412.     # --- Keterangan naratif ---
413.     ax_text = fig.add_subplot(gs[2, 2:])
414.     ax_text.axis("off")
415.     ax_text.text(0, 0.9, "Keterangan Otomatis:", fontsize=9.5,
416.                  fontweight="bold", color=WARN_AKSEN, va="top")
417.     ax_text.text(0, 0.55, result["keterangan"],
418.                  fontsize=9, color="#333333", va="top", wrap=True)
419.
420.     out_name = os.path.splitext(os.path.basename(filename))[
421.         0] + "_penjelasan.png"
422.     out_path = os.path.join(OUTPUT_DIR, out_name)
423.     plt.savefig(out_path, dpi=170, facecolor="white", bbox_inches="tight")
424.     plt.close()
425.     return out_path
426.
427. # =====
428. # 7. DASHBOARD RINGKASAN FOLDER
429. # =====
430. def save_folder_dashboard(results):
431.     n = len(results)
432.     n_sehat = sum(1 for r in results if r["label"] == "SEHAT")
433.     n_tidak = n - n_sehat
434.     rata_keyakinan = np.mean([r["confidence"] for r in results])
435.
436.     fig = plt.figure(figsize=(11, 6), facecolor="white")
437.     gs = GridSpec(2, 3, figure=fig, height_ratios=[
438.         0.3, 1], hspace=0.5, wspace=0.35)
439.
440.     ax_header = fig.add_subplot(gs[0, :])
441.     ax_header.axis("off")
442.     ax_header.text(0, 0.5, "DASHBOARD RINGKASAN PENGUJIAN DATASET",
443.                   fontsize=16, fontweight="bold", color=WARN_AKSEN)
444.     ax_header.axhline(y=0, xmin=0, xmax=1, color="#DDDDDD", linewidth=1.2)
445.
446.     # Kartu statistik
447.     kartu_info = [
448.         (str(n), "Total Gambar Diuji", WARN_AKSEN),
449.         (str(n_sehat), "Terklasifikasi Sehat", WARN_SEHAT),
450.         (str(n_tidak), "Terklasifikasi Tidak Sehat", WARN_TIDAK_SEHAT),
451.     ]
452.     for i, (angka, label, warna) in enumerate(kartu_info):
453.         ax = fig.add_subplot(gs[1, i])
454.         ax.axis("off")
455.         ax.set_xlim(0, 1)
456.         ax.set_ylim(0, 1)
457.         ax.add_patch(mpatches.FancyBboxPatch((0.05, 0.05), 0.9, 0.9,
458.                                                boxstyle="round,pad=0.02,rounding_
459. size=0.08",
460.                                                facecolor=warna, alpha=0.12,
461. edgecolor=warna, linewidth=1.5))
462.         ax.text(0.5, 0.62, angka, ha="center", va="center",
463.                fontsize=30, fontweight="bold", color=warna)
464.         ax.text(0.5, 0.28, label, ha="center", va="center",
465.                fontsize=10.5, color="#333333")
466.
467.     plt.tight_layout()
468.     out_path = os.path.join(OUTPUT_DIR, "dashboard_ringkasan.png")
469.     plt.savefig(out_path, dpi=170, facecolor="white", bbox_inches="tight")
470.     plt.close()
471.
472.     # --- Grafik distribusi keyakinan ---
473.     fig2, ax = plt.subplots(figsize=(7, 4.5), facecolor="white")

```

```

472.     labels_list = [r["label"] for r in results]
473.     conf_list = [r["confidence"] for r in results]
474.     colors_list = [WARNA_SEHAT if l ==
475.                    "SEHAT" else WARNA_TIDAK_SEHAT for l in labels_list]
476.     files_list = [os.path.splitext(r["file"])[0] for r in results]
477.
478.     bars = ax.barh(files_list, conf_list, color=colors_list)
479.     ax.set_xlabel("Tingkat Keyakinan (%)")
480.     ax.set_title("Tingkat Keyakinan Model per Gambar",
481.                  fontweight="bold", color=WARNA_AKSEN)
482.     ax.set_xlim(0, 105)
483.     for bar, conf in zip(bars, conf_list):
484.         ax.text(bar.get_width() + 1, bar.get_y() + bar.get_height() / 2,
485.                 f"conf: {conf:.1f}%", va="center", fontsize=8.5)
486.     plt.tight_layout()
487.     out_path2 = os.path.join(OUTPUT_DIR, "grafik_keyakinan_per_gambar.png")
488.     plt.savefig(out_path2, dpi=170, facecolor="white")
489.     plt.close()
490.
491.     return out_path, out_path2, n_sehat, n_tidak, rata_keyakinan
492.
493. # =====
494. # 8. MAIN
495. # =====
496. def run_single(image_path):
497.     result = predict_and_explain(image_path)
498.     out_path = save_explanation_figure(image_path, result)
499.
500.     print(f"\nGambar      : {image_path}")
501.     print(f"Hasil       : {result['label']}")
502.     print(f"Keyakinan   : {result['confidence']:.2f}%")
503.     print(f"Bercak      : {result['persen_bercak']:.2f}% area daun")
504.     print(f"Keterangan  : {result['keterangan']}")
505.     print(f"\n[INFO] Dashboard hasil disimpan di: {out_path}")
506.
507. def run_folder(folder_path):
508.     valid_ext = (".jpg", ".jpeg", ".png", ".bmp")
509.     results = []
510.     for filename in sorted(os.listdir(folder_path)):
511.         if not filename.lower().endswith(valid_ext):
512.             continue
513.         filepath = os.path.join(folder_path, filename)
514.         result = predict_and_explain(filepath)
515.         out_path = save_explanation_figure(filepath, result)
516.         result["file"] = filename
517.         results.append(result)
518.         print(f"{{filename:35s}} -> {{result['label']:12s}} "
519.               f"({result['confidence']:.2f}%, bercak "
520.               f"{result['persen_bercak']:.1f}%) "
521.               f"[disimpan: {out_path}])")
522.
523.     if not results:
524.         print(
525.             f"[PERINGATAN] Tidak ada gambar ditemukan di folder "
526.             f"'{folder_path}'."
527.         )
528.         return
529.
530.     dash_path, chart_path, n_sehat, n_tidak, rata = save_folder_dashboard(
531.         results)
532.     print("\n=== DASHBOARD RINGKASAN ===")
533.     print(
534.         f"Total      : {len(results)} | Sehat: {n_sehat} | Tidak Sehat: "
535.         f"{n_tidak}")
536.     print(f"Rata-rata keyakinan model: {rata:.2f}%")
537.     print(f"Dashboard   : {dash_path}")
538.     print(f"Grafik      : {chart_path}")

```

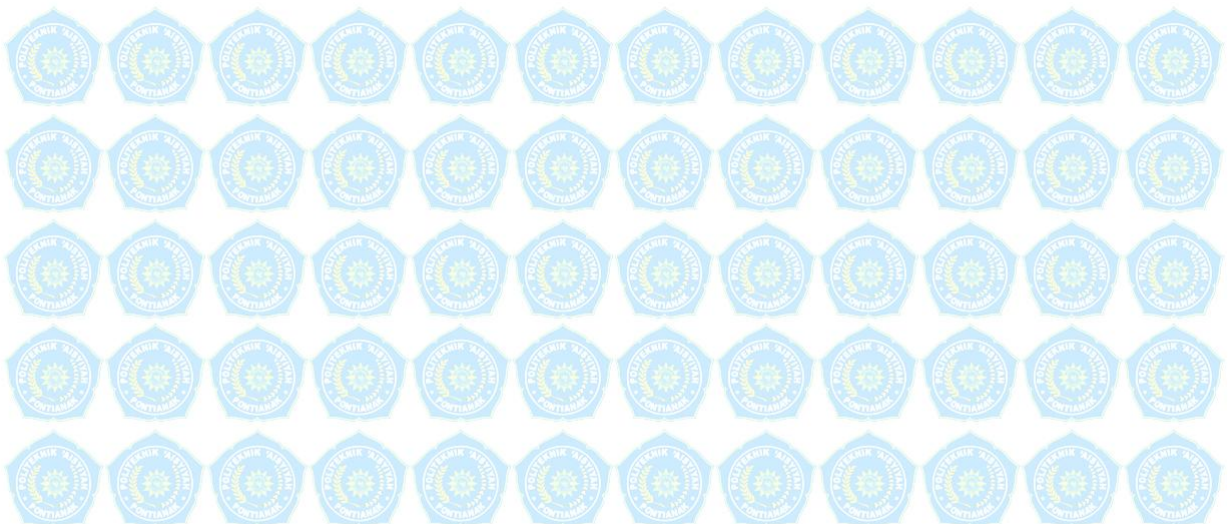
```

536. if __name__ == "__main__":
537.     parser = argparse.ArgumentParser(
538.         description="Prediksi + dashboard visual (Grad-CAM & deteksi bercak)
daun kratom")
539.     parser.add_argument("--image", type=str, help="Path ke satu file gambar")
540.     parser.add_argument("--folder", type=str,
541.         help="Path ke folder berisi banyak gambar")
542.     args = parser.parse_args()
543.
544.     if args.image:
545.         run_single(args.image)
546.     elif args.folder:
547.         run_folder(args.folder)
548.     else:
549.         print(
550.             "Gunakan salah satu argumen: --image <path_gambar> atau --folder
<path_folder>")

```

PERPUSTAKAAN

NPP. 6171052A2000001



POLITEKNIK 'AISYIYAH PONTIANAK