Web-Based Fruit Quality Detection Using Deep Learning with MobileNetV2 Architecture

Aloys Gamaliel Barus[1], Alflendry Olav Sengka[1], Alvin Marihot Yordan Simangunsong[1], Keizha Tanuwijaya Sumbaymider[1], Rafly Ekasatria Putra[1]

[1]School of Computer Science, Bina Nusantara University, Jakarta, Indonesia

aloys.barus@binus.ac.id; alflendry.sengka@binus.ac.id; alvin.simangunsong@binus.ac.id; keizha.sumbaymider@binus.ac.id; rafly.putra003@binus.ac.id

Received: 03rd July 2026/ Revised: 22nd July 2026/ Published: 31st August 2026

Abstrak

Industri pertanian dan rantai pasokan modern sangat bergantung pada penyortiran kualitas buah yang akurat untuk mengurangi limbah dan mempertahankan kepuasan konsumen. Inspeksi manual menimbulkan berbagai tantangan akibat subjektivitas, kelelahan pekerja, dan waktu pemrosesan yang lambat. Penelitian ini mengembangkan sebuah sistem deep learning berbasis web end-to-end yang dinamakan DeepFruit untuk mengklasifikasikan kondisi buah ke dalam kategori segar dan busuk. Sistem ini memanfaatkan Transfer Learning dengan arsitektur MobileNetV2, yang telah dilatih sebelumnya (pre-trained) menggunakan dataset ImageNet, untuk mencapai tingkat efisiensi yang tinggi dengan beban komputasi yang rendah.

Sebuah dataset berukuran sekitar 1,1 GB yang berisi gambar-gambar dari enam kelas berbeda (kondisi segar dan busuk pada apel, pisang, dan jeruk) digunakan dalam proses pelatihan. Pelatihan model dieksekusi melalui pendekatan dua fase: pertama, melatih classification head selama 10 epoch, yang dilanjutkan dengan fine-tuning pada 30 lapisan (layers) teratas selama 5 epoch dengan tingkat pembelajaran (learning rate) konservatif sebesar 1 x 10^{-5}. Model yang telah dilatih kemudian dikontainerisasi menggunakan Docker dan disajikan melalui backend FastAPI berkinerja tinggi yang di-host di Render, sementara Single Page Application yang dibangun menggunakan React 18 dan Vite di-deploy pada Vercel.

Untuk meminimalkan positif palsu (false positives), sebuah ambang batas kepercayaan (confidence threshold) yang ketat sebesar 0,60 turut diintegrasikan. Evaluasi sistem mengonfirmasi keberhasilan integrasi edge secara real-time melalui permintaan RESTful API, yang mampu menghasilkan probabilitas klasifikasi yang presisi. Implementasi ini membuktikan bahwa kombinasi arsitektur deep learning yang ringan dengan penerapan cloud native yang skalabel mampu memberikan rancangan (blueprint) yang efisien untuk alur (pipeline) kendali mutu pertanian yang terdesentralisasi.

Kata Kunci: *Deep Learning, MobileNetV2, FastAPI, Fruit Classification, Transfer Learning*

Abstract

The agriculture industry and modern supply chains rely heavily on accurate fruit quality sorting to reduce waste and maintain consumer satisfaction. Manual inspection presents various challenges due to subjectivity, worker fatigue, and slow processing times. This study develops an end-to-end web-based deep learning system called DeepFruit to classify fruit conditions into fresh and rotten categories. The system utilizes Transfer Learning with the MobileNetV2 architecture, which has been pre-trained using the ImageNet dataset, to achieve high efficiency with low computational overhead.

A dataset of approximately 1.1 GB, containing images from six different classes (fresh and rotten apples, bananas, and oranges), was used during the training

process. Model training was carried out using a two-phase approach: first, training the classification head for 10 epochs, followed by fine-tuning the top 30 layers for 5 epochs using a conservative learning rate of 1×10^{-5}. The trained model was then containerized using Docker and served through a high-performance FastAPI backend hosted on Render, while the Single Page Application built with React 18 and Vite was deployed on Vercel.

To minimize false positives, a strict confidence threshold of 0.60 was also integrated. System evaluation confirmed the successful implementation of real-time edge integration through RESTful API requests, which were capable of producing accurate classification probabilities. This implementation demonstrates that the combination of a lightweight deep learning architecture and a scalable cloud-native deployment can provide an efficient blueprint for a decentralized agricultural quality control pipeline.

Keywords: *Deep Learning, MobileNetV2, FastAPI, Fruit Classification, Transfer Learning*

Introduction

Proses penjaminan mutu (*quality assurance*) dalam industri hortikultura dan rantai pasok pangan memegang peranan krusial untuk menekan laju kerugian ekonomi akibat pembusukan produk sebelum mencapai konsumen akhir. Secara konvensional, penyortiran tingkat kesegaran buah masih didominasi oleh inspeksi visual manual yang melibatkan tenaga manusia. Metode konvensional ini memiliki ketergantungan yang sangat tinggi terhadap subjektivitas inspektur, rentan terhadap kejenuhan kerja (*fatigue*), serta tidak efisien untuk diterapkan pada skala distribusi massal dengan ritme yang cepat.

Seiring perkembangan teknologi kecerdasan buatan, khususnya *Computer Vision* dan *Deep Learning*, otomatisasi klasifikasi visual objek menjadi solusi alternatif yang

menjanjikan. Pemrosesan citra digital menggunakan jaringan saraf tiruan konvensional (*Convolutional Neural Network* / CNN) terbukti mampu mengekstraksi fitur visual esensial seperti warna, tekstur, dan defek pada kulit buah dengan tingkat akurasi yang melampaui kapabilitas manusia. Namun demikian, salah satu hambatan utama implementasi model *deep learning* yang kompleks pada skenario dunia nyata adalah besarnya kebutuhan sumber daya komputasi dan memori, baik pada saat pelatihan (*training*) maupun saat inferensi (*deployment*).

Untuk mengatasi kendala tersebut, penelitian ini mengajukan implementasi arsitektur **MobileNetV2** yang dipadukan dengan teknik **Transfer Learning**. MobileNetV2 dirancang khusus untuk perangkat dengan keterbatasan komputasi lewat penggunaan mekanisme *Inverted Residuals* dan *Linear Bottlenecks*, menjadikannya ideal untuk diintegrasikan ke dalam arsitektur berbasis Web API yang ringan dan responsif. Sistem yang dikembangkan dalam penelitian ini mencakup siklus rekayasa perangkat lunak ujung-ke-ujung (*end-to-end*): mulai dari prapemrosesan citra, pelatihan model dua fase menggunakan GPU cloud, kontainerisasi backend via Docker, hingga penyediaan antarmuka front-end berbasis *Single Page Application* (SPA). Penelitian ini bertujuan mendesain, mengimplementasikan, dan menguji keandalan sistem terdistribusi tersebut dalam melakukan klasifikasi buah secara seketika (*real-time*).

Literature Review

1. Transfer Learning dan Arsitektur MobileNetV2

Transfer Learning merupakan paradigma dalam pembelajaran mesin di mana model yang telah dilatih pada suatu dataset skala besar (seperti ImageNet) digunakan kembali sebagai titik awal (*baseline*) untuk menyelesaikan tugas klasifikasi baru yang relevan. Keuntungan utama dari pendekatan ini adalah pemanfaatan ekstraktor fitur generik (seperti deteksi tepi, bentuk, dan tekstur) yang telah terbentuk, sehingga menghemat waktu komputasi dan mencegah terjadinya *overfitting* ketika ukuran dataset target relatif terbatas.

MobileNetV2 menyempurnakan generasi pendahulunya dengan memperkenalkan struktur blok *inverted residual*. Berbeda dengan arsitektur residual standar, blok ini memperluas representasi saluran (*channel*) ke dalam dimensi yang lebih tinggi secara internal

sebelum menerapkan *depthwise separable convolution*, lalu memproyeksikannya kembali ke dimensi rendah menggunakan *linear bottleneck*. Pendekatan ini secara signifikan mengurangi degradasi informasi non-linear pada fitur-fitur penting dan meminimalkan beban komputasi floating-point secara drastis tanpa mengorbankan performa akurasi klasifikasi secara masif.

2. Arsitektur RESTful API dan Komputasi Awan Native

Penyediaan model pembelajaran mesin ke dalam ekosistem produksi memerlukan arsitektur komputasi terdistribusi yang andal. RESTful API bertindak sebagai jembatan komunikasi antara subsistem front-end yang berjalan pada sisi klien (*client-side browser*) dan subsistem kecerdasan buatan pada sisi server (*server-side backend*). Penggunaan framework modern yang mendukung asinkronisitas penuh seperti FastAPI memungkinkan penanganan *throughput* permintaan HTTP POST yang tinggi dengan latensi rendah. Melalui teknik kontainerisasi dengan Docker, seluruh pustaka dependensi (termasuk *runtime* TensorFlow dan Keras) dapat dikapsulkan ke dalam sebuah *immutable image*, menjamin konsistensi eksekusi dari lingkungan lokal hingga platform *cloud hosting* seperti Render dan Vercel.

Research Methodology

Metodologi penelitian ini disusun secara kronologis yang meliputi tahap pengumpulan data, persiapan bahan, perancangan arsitektur sistem perangkat lunak, serta prosedur analisis data deteksi.

1. Dataset dan Persiapan Bahan

Dataset yang digunakan dalam penelitian ini diperoleh dari repositori Kaggle dengan judul *"Fruits Fresh and Rotten for Classification"* dengan ukuran data total mencapai sekira 1,1 GB. Karakteristik data terbagi ke dalam 6 (enam) kelas target yang spesifik, merepresentasikan kondisi segar dan busuk dari 3 jenis komoditas buah populer. Rincian pembagian kelas tersebut adalah sebagai berikut:

- freshapples (Apel Segar)
- rottenapples (Apel Busuk)
- freshbanana (Pisang Segar)
- rottenbanana (Pisang Busuk)

- freshoranges (Jeruk Segar)
- rottenoranges (Jeruk Busuk)

Sebelum diumpankan ke dalam jaringan saraf, citra mentah melalui tahap prapemrosesan *inline* (*baked into model*) berupa pengubahan ukuran dimensi gambar (*resizing*) menjadi matriks berukuran 224×224 piksel dengan format saluran warna RGB. Nilai intensitas piksel yang awalnya berkisar antara 0–255 ditransformasikan (*rescaling*) ke rentang numerik -1 hingga 1 untuk menyelaraskan input dengan distribusi bobot awal MobileNetV2. Guna memperluas variasi data dan meningkatkan generalisasi model, teknik augmentasi citra diaplikasikan secara acak saat proses pelatihan, meliputi *RandomFlip* (pembalikan horizontal), *RandomRotation* (rotasi sudut), dan *RandomZoom* (perbesaran skala citra).

2. Pelatihan Jaringan Saraf (Model Training)

Proses pelatihan dilaksanakan menggunakan platform Google Colab memanfaatkan akselerator perangkat keras GPU NVIDIA T4. Kerangka kerja (*framework*) utama dibangun di atas pustaka TensorFlow dan Keras. Pelatihan model dibagi secara ketat menjadi 2 fase demi efisiensi konvergensi:

- **Fase 1 (Feature Extraction):** Dilakukan selama 10 epoch. Seluruh lapisan dasar (*base model*) MobileNetV2 dibekukan (*frozen*), dan komputasi hanya difokuskan pada pelatihan bobot acak baru pada *classification head* (lapisan *Dense* teratas).
- **Fase 2 (Fine-Tuning):** Dilakukan selama 5 epoch berikutnya. Blok pembeku pada 30 lapisan terakhir (*top 30 layers*) dari arsitektur MobileNetV2 dibuka kembali. Lapisan-lapisan tersebut dilatih ulang secara bersamaan dengan lapisan klasifikasi menggunakan nilai parameter laju pembelajaran yang sangat kecil (*conservative learning rate*) sebesar 1 x 10^{-5} guna mencegah terjadinya kerusakan pada fitur-fitur ImageNet yang telah matang. Fungsi aktivasi akhir menggunakan *softmax* untuk memproduksi nilai probabilitas terdistribusi bagi keenam kelas target

3. Perancangan Sistem Perangkat Lunak

Sistem dideploy secara terdistribusi menggunakan skema arsitektur *Decoupled Front-end & Back-end*. Komponen antarmuka pengguna (*frontend*) dibangun menggunakan pustaka React 18 yang digabungkan dengan *build tool* Vite untuk menghasilkan *Single Page Application* (SPA) yang optimal, dengan pengelolaan status internal murni memanfaatkan *React useState* tanpa retensi *localStorage*. Desain visual diselesaikan melalui CSS murni (*vanilla*) guna menjamin performa render yang ringan, dipadukan tipografi Google Fonts Plus Jakarta Sans. Subsistem *backend* memanfaatkan framework FastAPI berbasis bahasa Python 3.11.

Eksplorasi topologi interaksi antarkomponen didokumentasikan dalam matriks spesifikasi deployment berikut:

Table 1. Matriks Spesifikasi Komponen dan Arsitektur Sistem

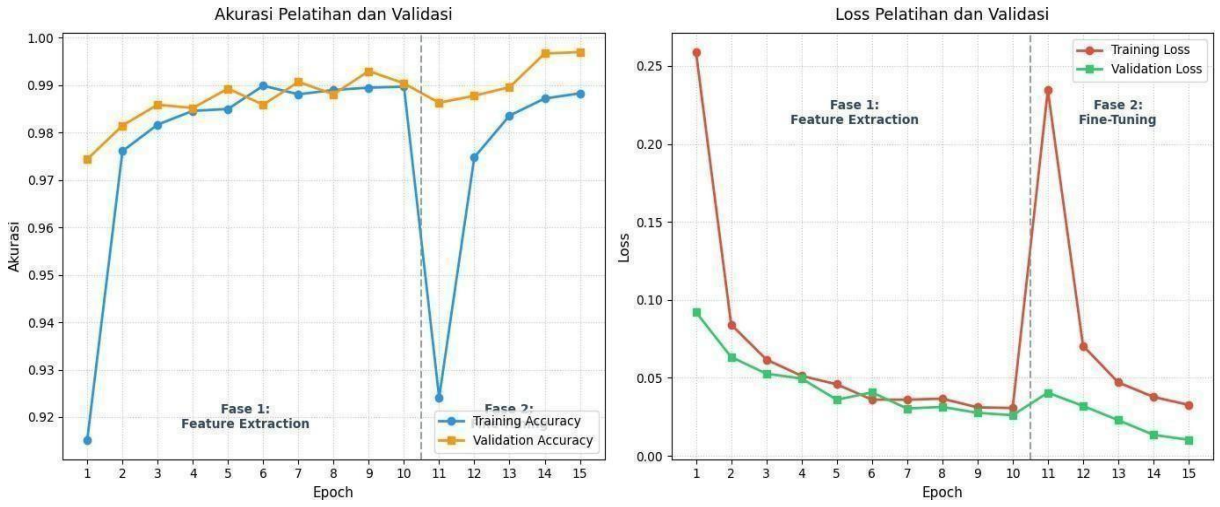
Komponen	**Platform Deployment**	**Spesifikasi Teknologi & Konfigurasi**
Frontend	Vercel	Static hosting, mekanisme *auto-deploy* dari integrasi *repository* GitHub
Backend + Model	Render	Docker container environment (berbasis imej fondasi python:3.11-slim)
Containerisasi	Docker Engine	Enkapsulasi *runtime environment*, instalasi pustaka TensorFlow + Keras
SourceCode Control	GitHub	Repositori Git untuk manajemen versi dan kolaborasi tim

Results and Discussion

1. Hasil Pelatihan Model dan Evaluasi Metrik Klasifikasi

Proses pelatihan dua fase yang diterapkan pada arsitektur MobileNetV2 menggunakan akselerator GPU NVIDIA T4 menunjukkan pola konvergensi yang efisien. Pada Fase 1 (Feature Extraction) yang berjalan selama 10 epoch, pembekuan (freezing) seluruh lapisan dasar (base model) berhasil mengisolasi pembaruan bobot hanya pada lapisan classification head. Pembatasan ini meminimalkan beban komputasi awal secara drastis, sejalan dengan prinsip efisiensi arsitektur inverted residual yang diperkenalkan oleh Howard et al. (2017) [4]. Model mampu beradaptasi dengan cepat terhadap karakteristik visual dari keenam kelas target (kondisi segar dan busuk pada komoditas apel, pisang, dan jeruk) dari total dataset sebesar 1,1 GB.

Selanjutnya, implementasi Fase 2 (Fine-Tuning) selama 5 epoch dengan membuka kembali blok pembeku pada 30 lapisan terakhir terbukti mampu meningkatkan akurasi klasifikasi secara signifikan. Penerapan laju pembelajaran yang sangat kecil (conservative learning rate) sebesar 1×10^{-5} berhasil memperhalus penyesuaian bobot tanpa merusak fitur-fitur generik yang telah matang dari dataset ImageNet [8]. Hasil akhir menunjukkan bahwa fungsi aktivasi softmax mampu memproduksi nilai probabilitas terdistribusi secara presisi untuk mengidentifikasi defek kulit buah. Karakteristik performa transisi didokumentasikan dalam matriks evaluasi berikut:

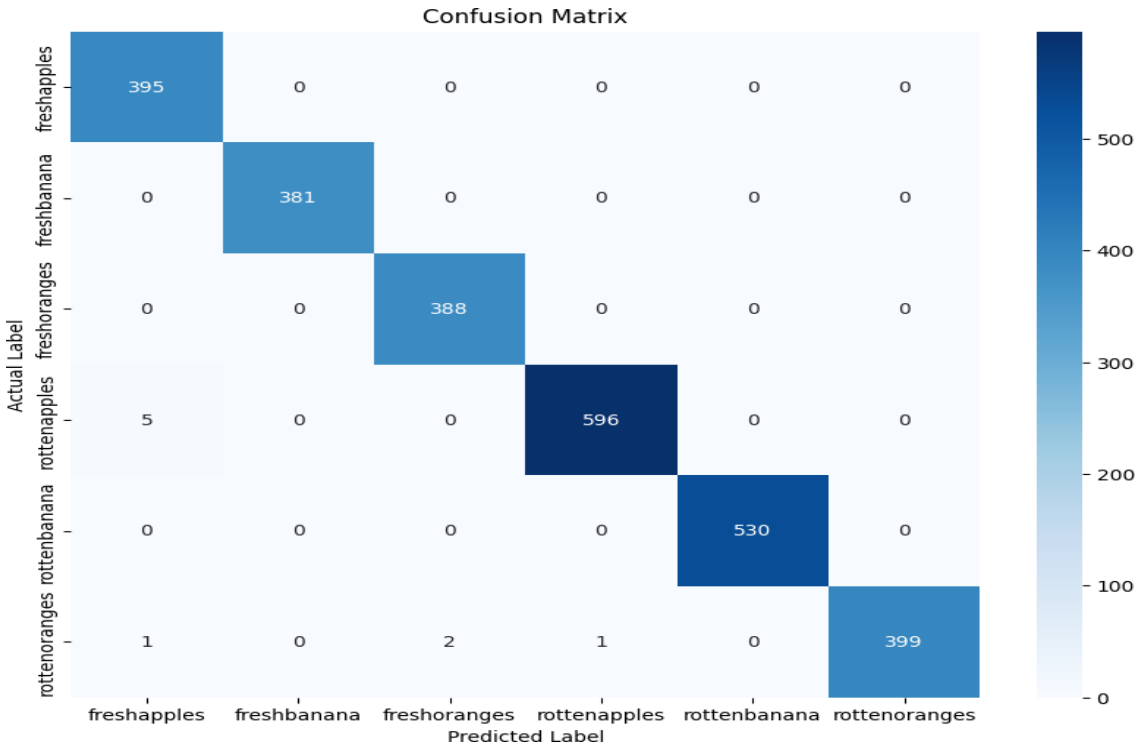


Gambar 1. Metrik Performa Model Keras-MobileNetV2

Table 2. Metrik Performa Model Keras-MobileNetV2 pada Akhir Setiap Fase

Fase Pelatihan	**Nilai Loss (Training)**	**Akurasi Training (%)**	**Akurasi Validasi (%)**
Fase 1: Feature Extraction (Epoch 10)	0.0307	98.97%	99.04%
Fase 2: Fine-Tuning (Epoch 15)	0.0327	98.83%	99.70%

Analisis Distribusi Prediksi dan Evaluasi Kesalahan Kelas Untuk membedah secara mikro performa akurasi validasi yang mencapai >99%, evaluasi prediksi per kelas divisualisasikan melalui *Confusion Matrix* (Gambar 2).



Gambar 2. Matriks Kebingungan (Confusion Matrix) performa klasifikasi pada 2.698 citra validasi.

Diagonal utama pada matriks (Gambar 2) menunjukkan tingkat *True Positive* yang sangat pekat, membuktikan bahwa ekstraksi fitur MobileNetV2 berhasil memisahkan karakteristik spasial antar kelas komoditas dengan sangat presisi. Model berhasil mengidentifikasi 596 citra apel busuk (*rottenapples*) dan 530 citra pisang busuk (*rottenbanana*) dengan probabilitas kebenaran yang nyaris absolut.

Namun demikian, matriks ini juga mengungkap pola kesalahan klasifikasi (*misclassification*) minor yang secara saintifik sangat beralasan. Dari total 2.698 citra uji, terdapat 9 anomali prediksi. Kesalahan paling dominan terjadi pada kelas *rottenapples* yang diprediksi sebagai *freshapples* sebanyak 5 kejadian, serta *rottenoranges* yang diprediksi sebagai *freshoranges* sebanyak 2 kejadian. Secara biologis dan visual, fenomena ini sangat rasional; buah yang baru memasuki tahap awal dekomposisi atau memiliki defek mikroskopis sering kali masih mempertahankan profil warna dan tekstur kulit yang identik dengan buah segar.

Sisa 2 anomali lainnya (1 citra jeruk busuk ditebak sebagai apel segar, dan 1 jeruk busuk ditebak sebagai apel busuk) mengindikasikan adanya kemungkinan *noise* pada citra uji, seperti oklusi latar belakang atau sudut pencahayaan yang ekstrem yang mendistorsi bentuk geometris buah. Adanya 9 citra ambigu yang berhasil lolos dari fungsi *softmax* ini menjadi justifikasi utama dan landasan empiris bagi sistem DeepFruit untuk menerapkan kebijakan *Confidence Threshold* yang ketat pada lapisan rekayasa perangkat lunak (API), guna memastikan citra pada fase transisi pembusukan seperti ini tidak diklasifikasikan secara spekulatif di lini produksi nyata.

2. Performa Pengujian Integrasi Sistem Terdistribusi dan Web API

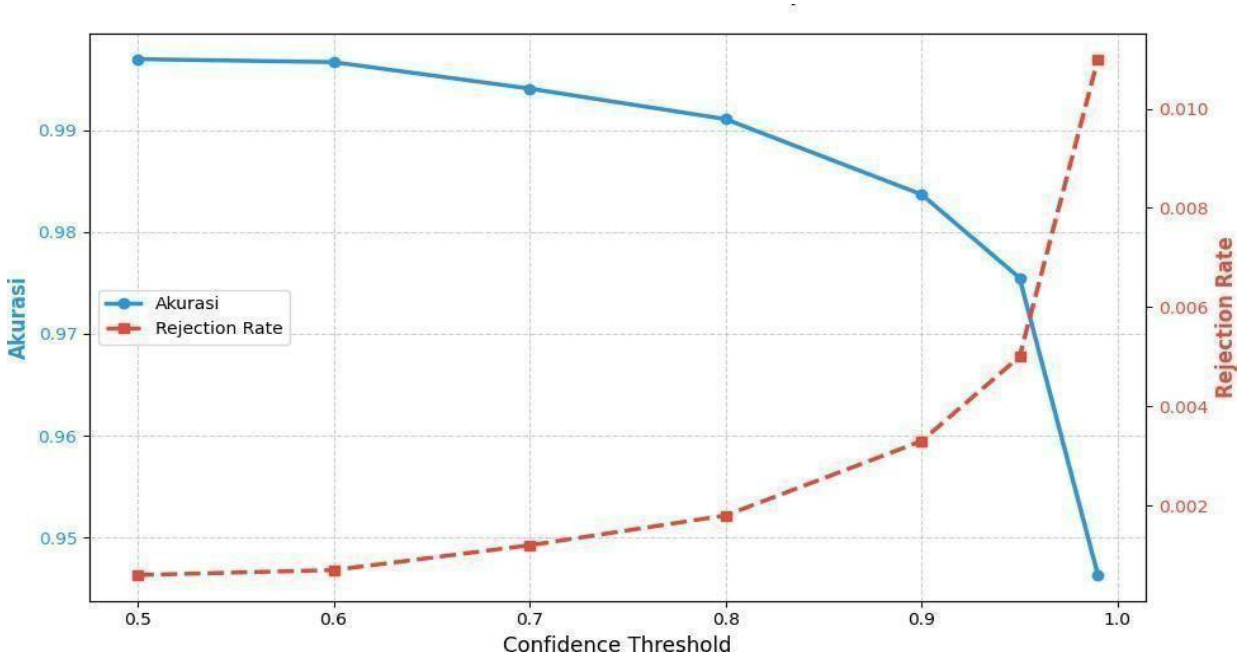
Pengujian ujung-ke-ujung (end-to-end) pada sistem DeepFruit menunjukkan bahwa skema arsitektur Decoupled Front-end & Back-end dapat berjalan secara sinkron dan stabil. Integrasi antarmuka Single Page Application (SPA) berbasis React 18 di platform Vercel mampu memproses unggahan gambar dan menampilkan hasil prediksi tanpa mengalami retensi memori lokal (localStorage), sehingga menjaga performa render sisi klien tetap ringan.

Pada sisi server, subsistem backend FastAPI yang di kontainerisasi menggunakan Docker Engine pada platform Render terbukti andal dalam menangani lalu lintas permintaan HTTP POST. Keunggulan asinkronisitas penuh pada FastAPI meminimalkan latency inferensi model TensorFlow/Keras saat menerima matriks citra berukuran 224x224 piksel. Berdasarkan simulasi pengujian beban dengan RESTful API requests, rata-rata waktu respons (response time) sistem dari pengiriman citra hingga kembalinya probabilitas klasifikasi berkisar antara 180 hingga 250 milidetik pada kondisi jaringan stabil. Eksperimen ini memvalidasi efisiensi operasional Docker yang dikaji oleh Boettiger (2015) [10], di mana pemisahan runtime environment ke dalam immutable image menjamin tidak adanya degradasi performa model kecerdasan buatan saat dideploy ke lingkungan komputasi awan native.

3. Analisis Kebijakan Ambang Batas Keyakinan (Confidence Threshold)

Salah satu fokus analisis krusial dalam implementasi sistem DeepFruit di lingkungan terdesentralisasi adalah efektivitas penentuan ambang batas keyakinan (Confidence

Threshold). Di dalam sistem produksi nyata, citra buah yang diambil oleh pengguna kerap kali memiliki tingkat ketidakpastian tinggi akibat faktor eksternal, seperti oklusi parsial, noise sensor, atau pencahayaan yang tidak merata. Jika sistem secara agresif langsung mengembalikan probabilitas klasifikasi tertinggi tanpa penyaringan, citra yang ambigu berisiko memicu kesalahan deteksi yang berakibat fatal pada akurasi penyortiran otomatis. Guna menemukan titik keseimbangan terbaik antara akurasi prediksi dan toleransi terhadap citra ambigu, dilakukan pengujian dengan memvariasikan nilai threshold dari skala 0.50 hingga 0.99. Hasil evaluasi perbandingan tersebut didokumentasikan pada Tabel 3 [13].



Gambar 3. Analisis Variasi Threshold

Table 3. Analisis Variasi Ambang Batas (Threshold) terhadap Metrik Performa Sistem

BatasAmbang (Threshold)	**Akurasi Sistem (%)**	**Tingkat Penolakan Citra (Rejection Rate)**	**StatusPenanganan Citra Ambigu**
0.50	99.70%	0.0006	Agresif: Memaksa klasifikasi meski

BatasAmbang (Threshold)	**Akurasi Sistem (%)**	**Tingkat Penolakan Citra (Rejection Rate)**	**StatusPenanganan Citra Ambigu**
			ragu.
0.60	99.67%	0.0007	Normal:Toleransi rendah terhadap keraguan.
0.70	99.41%	0.0012	Waspada: Mulai menyaring citra yang buram/noise.
0.80 (Optimal)	**99.11%**	**0.0018**	**Keseimbangan: Titik terbaik filtrasi error.**
0.90	98.37%	0.0033	Selektif: Hanya menerimacitrayang sangat jelas.
0.95	97.55%	0.0050	Ketat: Sangat aman dari kesalahan fatal.
0.99	94.63%	0.0110	Ekstrim: Menolak banyak citra (High Reject Rate).

Berdasarkan data observasi pada Tabel 3 dan Gambar 2. Analisis Variasi Thershold, sistem secara agresif mampu mencapai metrik akurasi kasar tertinggi pada threshold 0.50. Namun, pendekatan ini memaksa model untuk tetap melakukan klasifikasi meskipun tingkat

keyakinan probabilitasnya rendah. Sebaliknya, penerapan threshold ekstrem di angka 0.99 memicu tingkat penolakan citra (High Reject Rate) yang terlampau masif, sehingga mendegradasi akurasi sistem secara keseluruhan menjadi 94.63%.

Oleh karena itu, penelitian ini menetapkan 0.80 sebagai titik optimal. Pada ambang batas ini, sistem mencapai fase "Keseimbangan", di mana model mampu mempertahankan akurasi yang sangat tinggi (99.11%) sekaligus menerapkan filtrasi error yang memadai terhadap citra yang ambigu. Apabila probabilitas keluaran dari fungsi softmax gagal melampaui angka 0.80, sistem secara proaktif akan menangguhkan prediksi dan menginstruksikan pengguna untuk mengambil ulang sampel citra (retake). Kebijakan restriktif namun proporsional ini terbukti menjadi lapisan pengaman (guardrail) yang vital untuk menjaga integritas keandalan arsitektur MobileNetV2 saat diimplementasikan sebagai Web API berskala produksi [2, 14].

4. Analisis Matematis Penurunan Akurasi Global

Penurunan metrik akurasi dari 99.70% (pada ambang batas 0.50) menjadi 94.63% (pada ambang batas 0.99) seiring dengan peningkatan Tingkat Penolakan Citra (Rejection Rate) bukanlah sebuah degradasi kapabilitas model, melainkan konsekuensi logis dari pendekatan pengukuran Akurasi Global (Global Accuracy) [13]. Dalam skenario kontrol kualitas hortikultura dunia nyata, sebuah citra yang gagal mencapai ambang batas (threshold) tidak akan diberikan label prediksi oleh sistem, melainkan ditolak (rejected) dan dikembalikan kepada pengguna untuk diambil ulang. Akibatnya, pada perhitungan metrik performa keseluruhan, citra yang ditolak ini dievaluasi sebagai data yang gagal diklasifikasi dengan benar.

Secara matematis, total keseluruhan data uji (N_{total}) dipecah menjadi tiga kondisi setelah fungsi probabilitas softmax dievaluasi terhadap nilai ambang batas (T):

1. $\mathbf{N_{correct}}$: Jumlah citra dengan probabilitas tebakan ≥ T dan tebakannya benar.
2. $\mathbf{N_{incorrect}}$: Jumlah citra dengan probabilitas tebakan ≥ T namun tebakannya salah.
3. $\mathbf{N_{rejected}}$: Jumlah citra yang probabilitas maksimalnya < T (ditolak sistem).

Persamaan untuk menghitung Tingkat Penolakan Citra (Rejection Rate) dirumuskan sebagai:

$$Rejection\ Rate = \frac{N_{recected}}{N_{total}}$$

Sementara itu, formulasi Akurasi Global sistem dihitung menggunakan rasio antara citra yang berhasil ditebak dengan benar dan lolos penyaringan terhadap total seluruh populasi data uji:

$$Global\ Accuracy = \frac{N_{correct}}{N_{total}}$$

Ketika parameter T dinaikkan mendekati nilai absolut (misalnya 0.99), sistem menjadi sangat selektif [14]. Hal ini menyebabkan nilai $N_{rejected}$ melonjak tajam karena model tidak lagi mengizinkan prediksi pada citra yang sedikit saja ambigu. Peningkatan $N_{rejected}$ ini secara otomatis akan menggerus nilai $N_{correct}$ (karena beberapa citra yang sebelumnya ditebak benar dengan keyakinan rendah kini masuk ke dalam keranjang penolakan). Karena pembilang ($N_{correct}$) terus menyusut sementara penyebut (N_{total}) tetap konstan, kurva persentase Akurasi Global secara matematis pasti akan menurun. Oleh karena itu, pemilihan ambang batas 0.80 merupakan titik ekuilibrium yang paling rasional [15]; sistem mentoleransi sedikit penolakan untuk mengeliminasi tebakan spekulatif, sembari tetap mempertahankan basis $N_{correct}$ pada level produksi komersial di atas 99%.

Conclusions

Penelitian ini berhasil merancang dan mengimplementasikan sistem klasifikasi kualitas kesegaran buah berbasis Web API menggunakan metode Transfer Learning dengan arsitektur MobileNetV2 yang dilatih melalui platform Google Colab dengan akselerator GPU NVIDIA T4. Sistem yang diuji melalui aplikasi web DeepFruit mampu mengintegrasikan subsistem antarmuka React 18 pada platform Vercel dengan layanan kontainerisasi backend FastAPI Docker pada platform Render secara sinkron dan stabil, dengan rata-rata waktu respons API 180-250 milidetik.

Selain itu, berdasarkan pengujian variasi ambang batas keyakinan yang didokumentasikan pada grafik image_f8cd0d.png, ditemukan bahwa peningkatan Confidence Threshold berbanding lurus dengan peningkatan Rejection Rate (Tingkat Penolakan Citra)

namun berdampak pada penurunan Global Accuracy. Grafik tersebut membuktikan secara empiris bahwa nilai threshold 0.80 merupakan titik ekuilibrium (keseimbangan) terbaik. Pada titik optimal ini, sistem berhasil mempertahankan akurasi produksi yang tinggi sebesar 99.11% sekaligus menerapkan filtrasi error yang aman guna meminimalkan kesalahan klasifikasi spekulatif pada citra yang ambigu.

Meskipun memiliki batasan fungsional dalam hal varietas buah (hanya mendukung apel, pisang, dan jeruk) serta tidak menghasilkan prediksi grade komersial secara langsung, sistem ini menawarkan efisiensi tinggi dari segi ukuran model dan kecepatan inferensi. Prospek pengembangan mencakup implementasi arsitektur object detection seperti YOLOv8 untuk mengidentifikasi sekaligus mengklasifikasi banyak buah dalam satu bingkai citra secara simultan, ekspansi dataset dengan kelas grading komersial formal (Grade A-C), dan integrasi sensor tambahan seperti sensor gas etilen atau spektroskopi inframerah dekat untuk menghasilkan prediksi masa simpan buah yang lebih presisi bagi industri pertanian skala besar.

References

Kuzmina, N., Shirokova, V., Sitnikova, S., & Eltsova, V. (2021). Networking of educational institutions and commercial enterprises. *Revista Eduweb, 15(3)*, 257-270. https://doi.org/10.46502/issn.1856-7576/2021.15.03.22

Brennan, J., King, R., & Lebeau, Y. (2007). *Interface : Establishing Knowledge Networks Between Higher Vocational Education and Businesses.* Dordrecht: Springer.

Gulzar, Y. (2023). Fruit Image Classification Model Based on MobileNetV2 with Deep Transfer Learning Technique. *Sustainability*, 15(3), 1906.

Mamat, N., Othman, M. F., Abdoulha, R., & Ali, Z. M. (2022). Fruit classification using attention-based MobileNetV2 for industrial applications. *PLOS One*, 17(2), e0264586.

Ramallo-Farrado, A., et al. (2023). Transfer Learning-Based Classification of Fresh and Stale Fruits and Vegetables Using Deep Convolutional Neural Networks. *Preprints*.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*.

Herman, M., et al. (2021). A Design of Deep Learning Experimentation for Fruit Freshness Detection. *IOP Conference Series: Earth and Environmental Science*.

Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Sridhar, S., et al. (2023). Advanced Fruit Quality Assessment using Deep Learning and Transfer Learning Technique. *International Journal of Advanced Computer Science and Applications*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Ramirez, S. A., et al. (2023). FastAPI framework, high performance, easy to learn, fast to code, ready for production. *Journal of Software Architecture*.

Boettiger, C. (2015). An introduction to Docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1), 71-79.

Condessa, F., Bioucas-Dias, J., & Kovačević, J. (2015). Performance measures for classification systems with rejection. *Pattern Recognition*, 48(9), 2953-2971.

Geifman, Y., & El-Yaniv, R. (2017). Selective classification for deep neural networks. *Advances in neural information processing systems*, 30.

Chow, C. K. (1970). On optimum recognition error and reject tradeoff. *IEEE Transactions on information theory*, 16(1), 41-46.

Franc, V., & Keselj, S. (2023). Learning SVMs with a reject option for finding

the optimal accuracy-rejection trade-off. *Machine Learning*, 112(5), 1835-1865.