Djtechno : Jurnal Teknologi Informasi
Vol. 5, No. 2 Agustus 2024
E-ISSN: 2745-3758, P-ISSN : 2776-8546 DOI: 10.46576/djtechno

INTEGRASI MODEL DEEP LEARNING EFFICIENTNET-B0 UNTUK DETEKSI PENYAKIT DAUN TOMAT PADA APLIKASI SELULER BERBASIS FLUTTER

Taufiqurrahman[1], Indah Clara Sari [2], Micka Krisdayani Manurung[3]
1,2,3) Teknologi Rekayasa Perangkat Lunak, Politeknik Wilmar Bisnis Indonesia, Indonesia

Article Info

Article history:

Received: 14 Juli 2024
Revised: 24 Juli 2024
Accepted: 08 Agustus 2024

ABSTRACT

Abstrak

Penelitian ini bertujuan mengembangkan model Deep Learning EfficientNet-B0 untuk mendeteksi penyakit daun tomat dan mengintegrasikannya ke dalam aplikasi seluler berbasis Flutter. Metode penelitian mencakup pelatihan model menggunakan dataset gambar dengan teknik optimasi seperti quantization dan pruning untuk efisiensi pada perangkat dengan sumber daya terbatas. Hasil penelitian menunjukkan bahwa model memiliki nilai loss sebesar 0.0939 dan akurasi 0.9955 pada data pengujian, serta waktu deteksi berkisar antara 0.150 hingga 0.554 detik. Implementasi model dalam aplikasi memungkinkan petani mendeteksi penyakit daun tomat secara real-time dengan akurasi tinggi, mendukung praktik pertanian berkelanjutan. Aplikasi ini dirancang agar mudah digunakan, memungkinkan pengguna untuk mengambil gambar daun tomat yang dicurigai terkena penyakit dan mendapatkan hasil diagnosis secara cepat. Melalui teknik optimasi, model ini dapat berjalan efisien pada perangkat dengan sumber daya terbatas tanpa mengorbankan akurasi. Penelitian ini memberikan kontribusi signifikan dalam penerapan teknologi kecerdasan buatan untuk sektor pertanian, menawarkan solusi praktis dan inovatif untuk mendeteksi penyakit tanaman melalui perangkat seluler, dan berpotensi meningkatkan efisiensi serta efektivitas dalam manajemen penyakit tanaman tomat.

Kata Kunci: Deep Learning, EfficientNet-B0, Deteksi Penyakit Daun Tomat, Flutter, TensorFlow Lite.

Abstract

This research aims to develop the EfficientNet-B0 Deep Learning model for detecting diseases in tomato leaves and integrate it into a mobile application based on Flutter. The research method includes training the model using an image dataset with optimization techniques such as quantization and pruning for efficiency on resource-limited devices. The results show a loss value of 0.0939 and an accuracy of 0.9955 on test data, with detection times ranging from 0.150 to 0.554 seconds. The implementation in the application allows farmers to detect tomato leaf diseases in real-time with high accuracy, supporting sustainable agricultural practices. The application is designed to be user-friendly, enabling users to capture images of suspected diseased tomato leaves and obtain quick diagnostic results. Through optimization techniques, this model operates efficiently on resource-constrained devices without sacrificing accuracy. This research provides a significant contribution to the application of artificial intelligence technology in the agricultural sector, offering practical and innovative solutions for detecting plant diseases via mobile devices, and potentially enhancing efficiency and effectiveness in managing tomato plant diseases.

Keywords: Deep Learning, EfficientNet-B0, Tomato Leaf Disease Detection, Flutter, TensorFlow Lite.

Corresponding Author:
E-mail : taufiq@wbi.ac.id

1 PENDAHULUAN

Dalam era digital yang berkembang pesat, aplikasi kecerdasan buatan telah menjadi elemen kunci dalam berbagai aspek kehidupan manusia. Salah satu subdisiplin *Artificial Intelligence* yang muncul dengan sangat signifikan adalah *Deep Learning* (Pembelajaran Mendalam), yang telah mengubah cara kita memproses data kompleks dan mengambil keputusan. Metode ini memungkinkan sistem komputer untuk memahami serta menggunakan informasi yang terkandung dalam data yang umumnya besar dan kompleks (Buhrmester et al., 2021; Yang et al., 2021), sesuatu yang biasanya sulit diolah dengan metode konvensional.

Salah satu kemajuan signifikan dalam perkembangan *Deep Learning* adalah penggunaan arsitektur model yang dikenal sebagai EfficientNet-B0. Arsitektur ini dikenal karena akurasinya yang tinggi (Bhupendra et al., 2022; Chen et al., 2023). Hal ini menjadikan EfficientNet-B0 sangat relevan dalam aplikasi dengan sumber daya terbatas. *EfficientNet-B0* merupakan salah satu *pre-trained models CNN* yang diusulkan oleh Mingxing Tan dan Quoc V. Le dari Google Research pada tahun 2019 (Tan & Le, 2019), EfficientNet-B0 menggunakan teknik yang disebut koefisien majemuk untuk memperbesar model dengan cara yang sederhana namun efektif (Punuri et al., 2023).

Penyakit pada tanaman tomat adalah masalah utama dalam pertanian yang berdampak signifikan pada produktivitas dan kualitas hasil panen. Penyakit ini disebabkan oleh berbagai patogen seperti virus, bakteri, dan jamur yang menyerang bagian tanaman, termasuk daun, batang, dan buah. Infeksi pada daun dapat menghambat fotosintesis dan menyebabkan defoliasi, mengurangi hasil panen drastis, serta menyebarkan penyakit ke tanaman sehat di sekitarnya. Identifikasi dini sangat penting untuk tindakan preventif efektif, seperti penggunaan fungisida atau bakterisida, pemilihan varietas tahan penyakit, dan praktik budidaya yang baik. Tanpa deteksi dan pengelolaan yang tepat, penyakit tanaman tomat dapat menyebabkan kerugian ekonomi besar bagi petani dan mengancam ketahanan pangan. Oleh karena itu, solusi inovatif untuk deteksi dini dan akurat penyakit daun tomat sangat diperlukan guna mendukung keberlanjutan pertanian dan kesejahteraan petani.

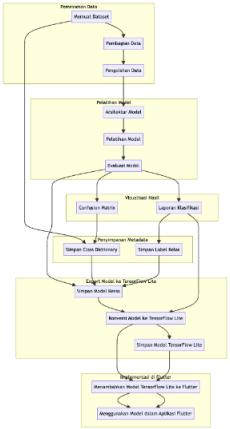
Dalam konteks ini, penggunaan teknologi pengenalan gambar berbasis *Deep Learning* menjanjikan sebagai solusi efektif. Dengan menganalisis citra daun tomat,

pada beberapa penelitian model *Deep Learning* dapat mengenali gejala penyakit daun dengan tingkat akurasi yang tinggi (Ridhovan & Suharso, 2022; Rozaqi et al., 2021; Windiawan et al., 2019). Dalam konteks yang lebih luas, ada kebutuhan mendesak untuk mengintegrasikan model *Deep Learning* yang telah terlatih dengan baik ke dalam aplikasi seluler. Langkah ini akan memberikan kemampuan kepada petani dan profesional pertanian untuk mendeteksi penyakit pada tanaman mereka dengan mudah menggunakan perangkat seluler. Salah satu pendekatan yang diusulkan adalah mengintegrasikan model *EfficientNet-B0* ke dalam aplikasi *Flutter* dengan menggunakan *TensorFlow Lite. TensorFlow Lite* adalah platform yang memungkinkan model *Deep Learning* berjalan efisien pada perangkat seluler yang memiliki sumber daya terbatas, seperti ponsel.

Kontribusi utama dari penelitian ini adalah penerapan model *EfficientNet-B0* yang unggul dalam klasifikasi gambar dan integrasinya ke dalam aplikasi *Flutter*, sehingga mudah diakses oleh pengguna akhir. Solusi yang diajukan memiliki tingkat akurasi tinggi dalam mengenali gejala penyakit pada daun tomat dan mempertimbangkan keterbatasan sumber daya perangkat seluler. Penelitian ini berpotensi menyediakan solusi efektif untuk mendukung pertanian berkelanjutan melalui pemanfaatan teknologi kecerdasan buatan, sehingga berkontribusi positif pada perkembangan aplikasi *Deep Learning* dalam pertanian modern.

2 METODE PENELITIAN

Penelitian ini melibatkan beberapa tahapan dalam proses klasifikasi penyakit daun pada tanaman tomat, yang mencakup berbagai metode dan teknik untuk mengidentifikasi jenis-jenis penyakit. Proses dimulai dengan pengumpulan data daun yang terinfeksi, analisis citra untuk mendeteksi gejala penyakit, dan penerapan algoritma klasifikasi untuk membedakan berbagai jenis penyakit.



Gambar 1 Diagram Penelitian.

Diagram penelitian ini menggambarkan alur kerja eksperimen dalam pengembangan model *Deep Learning* untuk pengenalan penyakit daun tomat. Langkah pertama adalah memuat dan memproses *dataset*, yang kemudian dibagi menjadi tiga subset: pelatihan, validasi, dan pengujian. Setelah itu, model *Deep Learning* dilatih dengan mendefinisikan arsitektur model dan menggunakan data pelatihan. Evaluasi model dilakukan menggunakan data validasi, dan hasil evaluasi seperti *confusion matrix* dan laporan klasifikasi ditampilkan.

Selanjutnya, model yang telah dilatih disimpan dalam format *Keras* dan dikonversi menjadi format *TensorFlow Lite* untuk penggunaan yang lebih efisien dalam aplikasi *mobile* (Veneman, 2021). Implementasi model *TensorFlow Lite* ke dalam aplikasi *Flutter* merupakan tahap terakhir dalam alur penelitian.

2.1 Dataset

Dataset yang digunakan pada penelitian ini adalah *dataset* yang diambil dari web kaggle.com memuat data gambar dengan masing-masing gambar terkait dengan suatu label klasifikasi tertentu. *Dataset* ini terdiri dari sebelas kategori klasifikasi yang mencakup berbagai kondisi dan penyakit yang mempengaruhi tanaman tomat, khususnya daun tomat. Kategori klasifikasi tersebut adalah sebagai berikut: *Mosaic_virus, Target_Spot, Bacterial_spot, Yellow_Leaf_Curl_Virus, Late_blight, Leaf_Mold, Early_blight, Spider_mites, Healthy*, dan *Septoria_leaf_spot*. Masing-masing kategori memiliki jumlah gambar sebanyak 1100, sehingga total keseluruhan dataset adalah 11000 gambar.

Tabel 1. *Tomato leaf disease dataset* .

Nama Label	**Jumlah Gambar**
Mosaic_virus	1100
Target_Spot	1100
Bacterial_spot	1100
Yellow_Leaf_Curl_Virus	1100
Late_blight	1100
Leaf_Mold	1100
Early_blight	1100
Spider_mites	1100
Healthy	1100
Septoria_leaf_spot	1100
Total	**11000**

2.2 Data Preprocessing

Pada tahap awal, dataset yang digunakan dalam penelitian ini diproses untuk mempersiapkan data latihan, validasi, dan pengujian. Tahapan ini dimulai dengan membagi dataset awal menjadi dua subset, yaitu data latih dan data sementara. Data latih memiliki proporsi sebesar 80% dari total *dataset,* sedangkan data sementara sebesar 20%. Selanjutnya, data sementara dibagi menjadi dua bagian lagi, yaitu data validasi dan data pengujian. Data validasi dan pengujian masing-masing memiliki

proporsi sebesar 60% dan 40% dari data sementara. Pembagian data ini penting untuk memastikan bahwa model *Deep Learning* dapat dipelajari dengan baik pada data latihan, dievaluasi dengan data validasi, dan diuji secara independen pada data pengujian.

Gambar 2 Sampel dari data latih.

Untuk memproses gambar-gambar dalam *dataset*, dilakukan pembuatan *generator* data menggunakan *ImageDataGenerator*. *Generator* data ini digunakan untuk melakukan augmentasi data pada gambar-gambar yang digunakan dalam pelatihan, sehingga dapat meningkatkan keberagaman dataset dan memperkuat generalisasi model (Chollet & others, 2016). Selain itu, *generator* data juga digunakan untuk memuat gambar-gambar dalam *batch* ke dalam model *Deep Learning*. *Batch size* sebesar 16 digunakan dalam proses pelatihan. Untuk data pengujian, ukuran *batch* disesuaikan dengan panjang data pengujian dengan batasan maksimum sebesar 80% dari panjang data pengujian. Hal ini diharapkan dapat mengoptimalkan penggunaan sumber daya komputasi dan mempercepat proses evaluasi model.

2.3 Arsitektur Model

Pada tahap ini, struktur model untuk pengenalan penyakit pada tanaman tomat dibangun. Pertama, ukuran citra (224x224 *pixel*) dan jumlah saluran warna (3 saluran) ditetapkan untuk membentuk dimensi citra yang sesuai. Selanjutnya, jumlah kelas yang akan diidentifikasi oleh model dihitung berdasarkan jumlah kunci kelas yang dihasilkan dari *generator* data pelatihan. Model ini menggunakan pendekatan transfer learning, memanfaatkan model *EfficientNet-B0* yang telah dilatih sebelumnya pada dataset *ImageNet*. *EfficientNet-B0* dipilih karena efisiensinya dan akurasi yang tinggi dalam pengolahan citra (Mahasin & Dewi, 2022).

Pada table 2 terdapat arsitektur model yang digunakan, model yang dibentuk terdiri dari beberapa lapisan, yaitu *EfficientNet-B0* sebagai lapisan dasar, lapisan *Batch Normalization* untuk normalisasi hasil dari lapisan sebelumnya, lapisan *Dense* dengan

256 unit dan aktivasi *ReLU* untuk memodelkan fitur-fitur kompleks, lapisan *Dropout* untuk mencegah *overfitting* dengan tingkat *dropout* sebesar 0,45, dan lapisan *Dense* terakhir dengan jumlah unit sesuai dengan jumlah kelas yang diidentifikasi, dan aktivasi *softmax* untuk menghasilkan probabilitas prediksi kelas. Setelah membangun arsitektur model, proses kompilasi dilakukan dengan menggunakan *Adamax optimizer* dengan *learning rate* 0,001, dan *categorical cross-entropy* sebagai fungsi *loss* untuk model klasifikasi multikelas. Metrik akurasi juga akan dihitung selama pelatihan.

Tabel 2 Arsitektur model yang digunakan

Layer	***Output Shape***	***Param***
Efficientnetb0 (Functional)	*(None, 1280)*	4049571
Batch Normalization	*(None, 1280)*	5120
Dense	*(None, 256)*	327936
Dropout	*(None, 256)*	0
Dense 1	*(None, 10)*	2570
Total params	***4385197 (16.73 MB)***	
Trainable params	***4340614 (16.56 MB)***	
Non-trainable params	***44583 (174.16 KB)***	

2.4 Pelatihan dan Evaluasi Model

2.4.1 Pelatihan Model

Pada tahap ini, model dikembangkan melalui proses pelatihan intensif menggunakan *dataset* yang telah diproses sebelumnya. Proses pelatihan melibatkan beberapa langkah kritis yang bertujuan untuk memperbarui bobot model melalui metode *backpropagation* dan optimisasi. Dalam penelitian ini, *hyperparameter* utama yang digunakan meliputi ukuran batch, jumlah epoch, dan laju pembelajaran. Ukuran *batch* mengacu pada jumlah sampel data yang diproses sebelum pembaruan *parameter* model dilakukan, sedangkan jumlah *epoch* menunjukkan jumlah siklus penuh melalui *dataset* pelatihan. Laju pembelajaran merupakan ukuran yang mengontrol seberapa besar perubahan bobot model pada setiap iterasi pembaruan.

Model *EfficientNet-B0* yang dimodifikasi dalam penelitian ini dilatih menggunakan *optimizer Adamax*, yang merupakan varian dari algoritma optimisasi Adam dengan beberapa modifikasi untuk stabilitas numerik yang lebih baik. Fungsi *loss* yang digunakan *adalah categorical cross-entropy*, yang merupakan fungsi *loss* standar untuk tugas klasifikasi multikelas (Fawwaz et al., 2020; Zuhan & Kristian, 2023). Selama proses pelatihan, metrik akurasi dan *loss* dari data pelatihan dan validasi dicatat pada setiap epoch untuk memantau konvergensi model. Dengan memantau metrik ini, kita dapat menentukan apakah model mengalami *overfitting* atau u*nderfitting*, dan menyesuaikan *hyperparameter* jika diperlukan.

2.4.2 Evaluasi Model

Setelah proses pelatihan selesai, evaluasi model dilakukan untuk mengukur performa sebenarnya pada data pengujian yang belum pernah dilihat oleh model

sebelumnya. Data pengujian ini digunakan untuk menilai generalisasi model terhadap data baru. Evaluasi model dilakukan dengan menggunakan beberapa metrik, termasuk akurasi, *precision, recall*, dan *F1-score* (Tangkelayuk, 2022).

Confusion matrix adalah tabel yang digunakan untuk menggambarkan kinerja model klasifikasi dengan membandingkan prediksi yang dilakukan oleh model dengan label sebenarnya . Tabel ini menunjukkan jumlah prediksi yang benar (*true positives* dan *true negatives*) serta jumlah prediksi yang salah (*false positives* dan *false negatives*) untuk setiap kelas. Dengan menggunakan *confusion matrix*, kita dapat menghitung metrik evaluasi lainnya seperti akurasi, *precision*, *recall*, dan F1-*score*.

1. **Akurasi (*Accuracy*):** Mengukur persentase prediksi yang benar dari total prediksi yang dilakukan oleh model.
$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN}$$
2. ***Precision*:** Mengukur proporsi prediksi positif yang benar dari semua prediksi positif yang dilakukan oleh model.
$$Precision = \frac{TP}{TP + FP}$$
3. ***Recall*:** Mengukur proporsi total kasus positif yang berhasil dideteksi oleh model.
$$Recall = \frac{TP}{TP + FN}$$
4. **F1-*score*:** Metrik harmonik yang menggabungkan precision dan recall, memberikan gambaran yang lebih komprehensif tentang performa model, terutama ketika ada ketidakseimbangan kelas dalam dataset.
$$F1 - score = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall}\right)$$
    - TP = *True Positives*
    - TN = *True Negatives*
    - FP = *False Positives*
    - FN = *False Negatives*

Dengan menggunakan metrik ini, kita dapat mengevaluasi seberapa baik model kita dalam melakukan klasifikasi pada data pengujian.

2.5 Implementasi Model ke *TensorFlow Lite*

Setelah model dilatih dan dievaluasi, langkah berikutnya adalah mengonversi model ke format *TensorFlow Lite* (TFLite). *TensorFlow Lite* merupakan framework yang dirancang untuk menjalankan model *TensorFlow* pada perangkat dengan sumber daya terbatas, seperti ponsel cerdas dan perangkat edge lainnya (Fajri et al., 2023).

Proses konversi ini melibatkan optimasi model untuk memastikan bahwa model dapat berjalan dengan efisien tanpa mengorbankan akurasi secara signifikan.

Proses konversi dimulai dengan menyimpan model *Keras* yang telah dilatih ke dalam format yang memungkinkan penyimpanan lengkap arsitektur model, bobot, dan konfigurasi pelatihan. Model ini kemudian dimuat kembali dan dikonversi ke format *TensorFlow Lite.* Konversi ini menghasilkan representasi *TensorFlow Lite* dari model yang siap untuk dioptimalkan. Optimasi model ini sangat penting untuk memastikan bahwa model dapat berjalan dengan efisien pada perangkat dengan sumber daya terbatas. Teknik optimasi yang digunakan meliputi *quantization* dan *pruning*. *Quantization* mengonversi bobot model dari float32 ke int8, yang secara signifikan mengurangi ukuran model tanpa penurunan akurasi yang signifikan. *Pruning*, di sisi lain, menghapus bobot yang tidak penting dalam model untuk mengurangi ukuran model lebih lanjut (Liang et al., 2021).

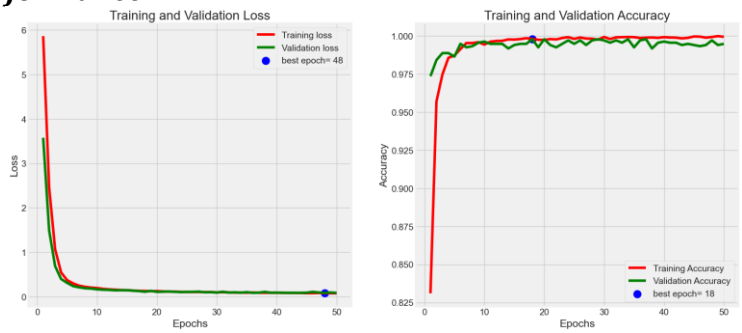
2.6 Implementasi Model ke Aplikasi Flutter

Flutter adalah *framework open-source* yang dikembangkan oleh *Google* untuk membangun aplikasi *natively compiled* untuk *mobile* (*iOS* dan *Android*), situs, dan *desktop* dari satu basis kode (Aulia Sabril, 2023; Yandika et al., 2023). *Flutte*r menggunakan bahasa pemrograman *Dart* dan menawarkan pendekatan reaktif dan deklaratif untuk pembangunan antarmuka pengguna. *Flutter* memungkinkan pengembang untuk membuat antarmuka pengguna yang cepat, indah, dan responsif dengan efisiensi tinggi dan performa yang mendekati aplikasi *native*.

Setelah model *TensorFlow Lite* dikonversi dan dioptimalkan, langkah selanjutnya adalah mengintegrasikannya ke dalam aplikasi *Flutter*. Proses ini memastikan model dapat digunakan secara efektif di aplikasi seluler. Pengujian dilakukan untuk memastikan model berjalan dengan baik dan memberikan prediksi akurat pada berbagai perangkat seluler, serta melibatkan pengguna akhir seperti petani untuk memastikan kemudahan penggunaan dan interpretasi hasil. Antarmuka pengguna dirancang intuitif, memungkinkan pengguna mengambil gambar daun tomat yang dicurigai terkena penyakit dan mendapatkan diagnosis secara *real-time*. Selain itu, antarmuka menyediakan informasi tambahan tentang gejala penyakit dan rekomendasi tindakan yang dapat diambil.

3. HASIL DAN PEMBAHASAN

3.1 Model Performance



Gambar 3 Grafik pelatihan dan validasi untuk loss dan akurasi

Grafik di sebelah kiri menunjukkan perubahan nilai *loss* selama pelatihan *(Training Loss)* dan validasi *(Validation Loss)* terhadap jumlah *epoch*. Pada awal pelatihan, nilai *loss* tinggi, menunjukkan kesalahan prediksi besar. Dengan bertambahnya *epoch*, nilai *loss* menurun tajam, menandakan model belajar dengan baik. Setelah sekitar 10 *epoch*, nilai *loss* stabil dan rendah, menunjukkan konvergensi. *Best epoch* untuk *loss* adalah pada *epoch* ke-48, dengan titik biru menunjukkan nilai terendah pada data validasi. Grafik di sebelah kanan menunjukkan perubahan nilai akurasi selama pelatihan *(Training Accuracy)* dan validasi *(Validation Accuracy)* terhadap jumlah *epoch*. Pada awalnya, nilai akurasi rendah, namun meningkat tajam seiring bertambahnya *epoch*, menunjukkan peningkatan kemampuan model dalam prediksi yang benar. Akurasi pelatihan mendekati 100% setelah sekitar 10 *epoch*, sementara akurasi validasi juga meningkat baik, meskipun sedikit lebih rendah, menunjukkan model generalisasi dengan baik. Best epoch untuk akurasi adalah pada epoch ke-18, dengan titik biru menunjukkan nilai tertinggi pada data validasi.

3.2 Evaluasi Model

```
11/11 [==============================] - 2s 151ms/step - loss: 0.0684 - accuracy: 1.0000
11/11 [==============================] - 2s 150ms/step - loss: 0.1118 - accuracy: 0.9830
11/11 [==============================] - 9s 766ms/step - loss: 0.0939 - accuracy: 0.9955
Train Loss:  0.06836508959531784
Train Accuracy:  1.0
--------------------
Validation Loss:  0.11178712546825409
Validation Accuracy:  0.9829545617103577
--------------------
Test Loss:  0.0939139723777771
Test Accuracy:  0.9954545497894287
```

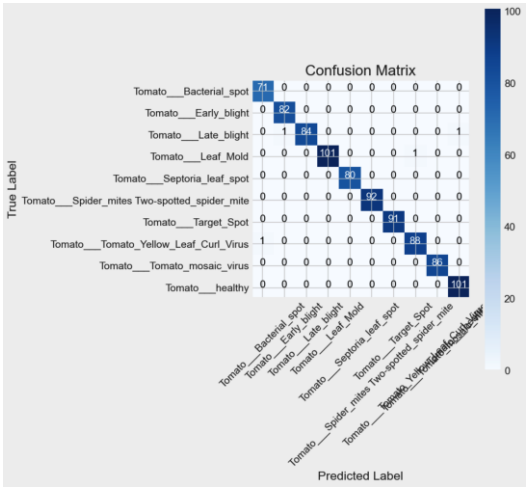
Gambar 4 Hasil evaluasi model

Setelah pelatihan, model dievaluasi menggunakan data pelatihan, validasi, dan pengujian. Pada data pelatihan, model menunjukkan nilai loss sebesar 0.0684 dan akurasi sempurna sebesar 1.0, menandakan model mempelajari pola dengan sangat baik. Pada data validasi, model memiliki nilai loss sebesar 0.1118 dan akurasi sebesar

0.9830, menunjukkan kemampuan generalisasi yang baik. Pada data pengujian, model mencapai nilai *loss* sebesar 0.0939 dan akurasi sebesar 0.9955, menandakan performa yang sangat baik pada data baru. Secara keseluruhan, model menunjukkan performa yang sangat baik dalam mengklasifikasikan penyakit pada daun tomat dengan akurasi tinggi dan kemampuan generalisasi yang kuat pada ketiga *subset* data.

3.3 Analisis *Confusion Matrix*

Pada Gambar 6, model menunjukkan performa sangat baik dalam mengklasifikasikan sebagian besar kelas penyakit. Sebagian besar nilai berada pada diagonal, menunjukkan prediksi benar untuk setiap kelas. Misalnya, dari 71 *instance* "*Bacterial spot*" dan 82 *instance* "*Early blight*", model memprediksi semuanya dengan benar.



Gambar 5 Plot hasil confusion matrix

Namun, ada beberapa kesalahan klasifikasi minor yang dapat dilihat dari sel-sel yang tidak berada pada diagonal. Sebagai contoh, terdapat satu *instance* dari kelas "*Late blight*" yang diklasifikasikan sebagai "Yellow Leaf Curl Virus".

	precision	recall	f1-score	support
Tomato___Bacterial_spot	0.99	1.00	0.99	71
Tomato___Early_blight	0.99	1.00	0.99	82
Tomato___Late_blight	1.00	0.98	0.99	86
Tomato___Leaf_Mold	1.00	0.99	1.00	102
Tomato___Septoria_leaf_spot	1.00	1.00	1.00	80
Tomato___Spider_mites Two-spotted_spider_mite	1.00	1.00	1.00	92
Tomato___Target_Spot	1.00	1.00	1.00	91
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.99	0.99	0.99	89
Tomato___Tomato_mosaic_virus	1.00	1.00	1.00	86
Tomato___healthy	0.99	1.00	1.00	101
accuracy			1.00	880
macro avg	1.00	1.00	1.00	880
weighted avg	1.00	1.00	1.00	880

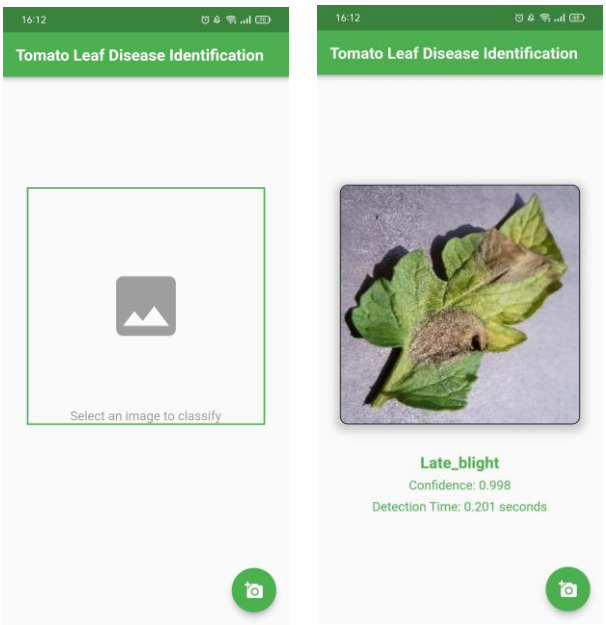
Gambar 6 Hasil confusion matrix yang dilakukan terhadap model

Precision mengukur ketepatan prediksi positif yang dilakukan oleh model, dan semua kelas memiliki precision sangat tinggi, mendekati atau mencapai 1.0. *Recall*

mengukur kemampuan model untuk menemukan semua instance positif, dan model ini memiliki *recall* sangat tinggi untuk semua kelas, menunjukkan hampir semua *instance* yang benar ditemukan oleh model. Semua kelas memiliki *F1-score* sangat tinggi, menandakan keseimbangan baik antara *precision* dan *recall*. Dengan akurasi keseluruhan 1.00 pada data pengujian yang terdiri dari 880 *instance*, model menunjukkan performa sangat baik dalam mendeteksi penyakit daun tomat. Metrik rata-rata makro dan tertimbang mendukung kesimpulan ini, dengan nilai mendekati atau mencapai 1.00 untuk *precision, recall*, dan *F1-score*.

3.4 Tampilan Aplikasi *Flutter*

Setelah mengembangkan aplikasi menggunakan Flutter, peneliti menguji aplikasi ini untuk memastikan bahwa model TensorFlow Lite yang telah dikonversi dapat berjalan dengan baik dan memberikan prediksi yang akurat dalam lingkungan mobile.



Gambar 8 Tampilan aplikasi flutter Tomato Leaf Disease Identification

Gambar di atas menunjukkan antarmuka pengguna dari aplikasi "Tomato Leaf Disease Identification". Pada tampilan awal, terdapat judul aplikasi dengan latar belakang hijau dan area untuk menampilkan gambar daun tomat yang akan diidentifikasi, dengan teks "*Select an image to classify*". Saat pengujian, gambar daun tomat yang terinfeksi ditampilkan dalam area persegi dengan border hijau. Di bawah gambar, terdapat informasi hasil identifikasi oleh model *TensorFlow Lite*, termasuk nama penyakit "***Late_blight***" dengan tingkat kepercayaan 0.998 dan waktu inferensi 0.201 detik.

3.5 Pengujian Aplikasi pada Berbagai Perangkat

Pengujian aplikasi "*Tomato Leaf Disease Identification*" dilakukan menggunakan berbagai perangkat dengan spesifikasi yang berbeda untuk memastikan konsistensi dan keandalannya. Perangkat yang digunakan dalam pengujian ini mencakup baik perangkat fisik maupun emulator. Perangkat fisik yang digunakan antara lain *Mediatek Helio P95* dengan *RAM 8 GB* (1), *Mediatek Helio P35* dengan *RAM 4 GB* (2), *Mediatek MT6765G Helio G35* dengan *RAM 4 GB* (3), dan *Snapdragon 712 AIE* dengan *RAM 4 GB* (4). Selain itu, pengujian juga dilakukan pada perangkat emulasi, yaitu *Android Emulated Device* (5) dan *iPhone Emulated Device* (6), keduanya dijalankan pada *Mac M2*. Penggunaan berbagai jenis perangkat ini memungkinkan pengujian kinerja aplikasi dalam berbagai konfigurasi perangkat keras, baik dalam lingkungan nyata maupun simulasi, sehingga memastikan bahwa aplikasi dapat beroperasi secara efisien dan andal dalam berbagai situasi operasional.

3.5.1 Tingkat *Confidence* Model pada Berbagai Perangkat

Hasil pengujian menunjukkan bahwa tingkat kepercayaan model sangat konsisten di berbagai perangkat, dengan nilai yang mendekati atau mencapai 1.000 untuk sebagian besar penyakit.

Tabel 3 Perbandingan Tingkat Kepercayaan *(Confidence)* Model pada Berbagai Perangkat

Label	*Confidence*					
	1	2	3	4	5	6
Early Blight	0,999	0,999	0,999	0,999	0,999	0,999
Healthy	0,999	0,999	0,999	0,999	0,999	0,999
Late Blight	0,998	0,998	0,998	0,998	0,998	0,998
Leaf Mold	1,000	1,000	1,000	1,000	1,000	1,000
Septoria	1,000	1,000	1,000	1,000	1,000	1,000
Spider Mites	0,999	0,999	0,999	0,999	0,999	0,999
Target Spot	1,000	1,000	1,000	1,000	1,000	1,000
Yellow Leaf Curl Virus	1,000	1,000	1,000	1,000	1,000	1,000
Mosaic Virus	1,000	1,000	1,000	1,000	1,000	1,000
Bacterial Spot	0,999	0,999	0,999	0,999	0,999	0,999

Penyakit seperti *"Leaf Mold", "Septoria", "Target Spot", "Yellow Leaf Curl Virus"*, dan *"Mosaic Virus"* memiliki tingkat kepercayaan sempurna (1.000) di semua perangkat yang diuji. Penyakit lainnya seperti *"Early Blight", "Healthy", "Spider Mites"*, dan *"Bacterial Spot"* memiliki tingkat kepercayaan sebesar 0.999, yang masih menunjukkan tingkat keyakinan yang sangat tinggi dari model. Hasil pengujian menunjukkan bahwa aplikasi *"Tomato Leaf Disease Identification"* mampu mendeteksi

penyakit daun tomat dengan tingkat kepercayaan yang sangat tinggi di berbagai perangkat.

Konsistensi tingkat kepercayaan ini mengindikasikan bahwa model *TensorFlow Lite* yang digunakan dalam aplikasi ini dapat diandalkan untuk memberikan hasil yang akurat di berbagai kondisi dan spesifikasi perangkat. Baik pada perangkat fisik dengan berbagai *chipset* dan *RAM,* maupun pada perangkat emulasi, model menunjukkan kinerja yang stabil dan sangat akurat, membuatnya cocok untuk digunakan oleh petani dan profesional pertanian di lapangan.

3.5.2 Waktu Deteksi Model pada Berbagai Perangkat

Hasil pengujian menunjukkan bahwa waktu deteksi model bervariasi tergantung pada spesifikasi perangkat, namun tetap dalam rentang yang dapat diterima untuk penggunaan praktis.

Tabel 4 Perbandingan Waktu Deteksi (*Detection Time)* Model pada Berbagai Perangkat

Label	***Detection Time (s)***					
	1	**2**	**3**	**4**	**5**	**6**
Early Blight	0,222	0,545	0,486	0,233	0,267	0,166
Healthy	0,214	0,474	0,479	0,192	0,391	0,165
Late Blight	0,223	0,474	0,494	0,214	0,267	0,155
Leaf Mold	0,209	0,490	0,491	0,194	0,293	0,168
Septoria	0,220	0,542	0,475	0,195	0,293	0,279
Spider Mites	0,216	0,477	0,498	0,202	0,429	0,153
Target Spot	0,211	0,481	0,480	0,191	0,209	0,165
Yellow Leaf Curl Virus	0,250	0,477	0,554	0,273	0,312	0,150
Mosaic Virus	0,205	0,510	0,474	0,202	0,325	0,170
Bacterial Spot	0,200	0,531	0,480	0,242	0,402	0,162

Waktu deteksi bervariasi tergantung pada spesifikasi perangkat yang digunakan. Perangkat dengan spesifikasi lebih tinggi, seperti *Mediatek Helio P95* dan *Snapdragon 712 AIE*, menunjukkan waktu deteksi yang lebih cepat. Waktu deteksi pada perangkat emulasi (baik *Android* maupun *iPhone*) umumnya sedikit lebih lama dibandingkan perangkat fisik, namun tetap dalam rentang yang dapat diterima untuk penggunaan praktis. Penyakit seperti *"Early Blight", "Healthy", "Late Blight"*, dan *"Spider Mites"* menunjukkan waktu deteksi yang cepat di semua perangkat. Waktu deteksi terendah adalah 0.150 detik untuk *"Yellow Leaf Curl Virus"* pada *iPhone Emulated Device*, sementara waktu deteksi tertinggi adalah 0.554 detik untuk *"Yellow Leaf Curl Virus"* pada *Mediatek MT6765G Helio G35.*

4. SIMPULAN

Penelitian ini berhasil mengembangkan dan mengintegrasikan model *Deep Learning EfficientNet-B0* untuk deteksi penyakit pada daun tomat ke dalam aplikasi seluler berbasis *Flutter*. Model ini menunjukkan akurasi yang sangat tinggi dalam mengidentifikasi berbagai penyakit daun tomat, seperti *Mosaic Virus, Target Spot,* dan *Yellow Leaf Curl Virus*, dengan tingkat kepercayaan yang hampir sempurna di berbagai perangkat fisik dan emulasi. Melalui teknik optimasi seperti *quantization* dan *pruning*, model ini dapat berjalan efisien pada perangkat dengan sumber daya terbatas tanpa mengorbankan akurasi. Hasil evaluasi menunjukkan bahwa model memiliki nilai *loss* sebesar 0.0939 dan akurasi 0.9955 pada data pengujian, menunjukkan kemampuan generalisasi yang kuat dan stabilitas performa yang baik. Selain itu, waktu deteksi model berkisar antara 0.150 hingga 0.554 detik tergantung pada spesifikasi perangkat, yang tetap dalam rentang yang dapat diterima untuk penggunaan praktis. Implementasi aplikasi ini memungkinkan petani dan profesional pertanian untuk mendeteksi penyakit daun tomat dengan mudah dan cepat, mendukung praktik pertanian berkelanjutan. Aplikasi *"Tomato Leaf Disease Identification"* menyediakan antarmuka yang intuitif dan mudah digunakan, memungkinkan pengguna untuk mengambil gambar daun tomat yang dicurigai terkena penyakit dan mendapatkan hasil diagnosis secara *real-time*. Dengan demikian, penelitian ini memberikan kontribusi signifikan dalam penerapan teknologi kecerdasan buatan untuk sektor pertanian, menawarkan solusi praktis dan inovatif untuk mendeteksi penyakit tanaman melalui perangkat seluler, dan berpotensi meningkatkan efisiensi serta efektivitas dalam manajemen penyakit tanaman tomat.

PUSTAKA

Aulia Sabril. (2023). Rancang bangun perangkat lunak antarmuka kendali mikrokontroler ESP826 dengan jaringan internet menggunakan flutter 3.0. *Micronic: Journal of Multidisciplinary Electrical and Electronics Engineering*, *8698*, 27–34. https://doi.org/10.61220/micronic.v1i2.2021

Bhupendra, Moses, K., Miglani, A., & Kumar Kankar, P. (2022). Deep CNN-based damage classification of milled rice grains using a high-magnification image dataset. *Computers and Electronics in Agriculture*, *195*, 106811. https://doi.org/https://doi.org/10.1016/j.compag.2022.106811

Buhrmester, V., Münch, D., & Arens, M. (2021). Analysis of Explainers of Black Box Deep Neural Networks for Computer Vision: A Survey. *Machine Learning and Knowledge Extraction*, *3*(4), 966–989. https://doi.org/10.3390/make3040048

Chen, X., Pu, X., Chen, Z., Li, L., Zhao, K.-N., Liu, H., & Zhu, H. (2023). Application of EfficientNet-B0 and GRU-based deep learning on classifying the colposcopy diagnosis of precancerous cervical lesions. *Cancer Medicine*, *12*(7), 8690–8699. https://doi.org/https://doi.org/10.1002/cam4.5581

Chollet, F., & others. (2016). Building powerful image classification models using very little data. *Keras Blog*, *5*, 90–95.

Fajri, R., Fitria, F., & others. (2023). Pengembangan Real-Time Object Detection System pada Perangkat Single-Board Computer. *KLIK: Kajian Ilmiah Informatika Dan Komputer*, *4*(2), 1154–1162.

Fawwaz, M. A. A., Ramadhani, K. N., & Sthevani, F. (2020). *Klasifikasi Ras pada hewan peliharaan menggunakan Algoritma Convolutional Neural Network (CNN)*. *8*(1), 715–730.

Liang, T., Glossner, J., Wang, L., Shi, S., & Zhang, X. (2021). Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, *461*, 370–403.

Mahasin, M., & Dewi, I. A. (2022). Comparison of CSPDarkNet53, CSPResNeXt-50, and EfficientNet-B0 Backbones on YOLO V4 as Object Detector. *International Journal of Engineering, Science and Information Technology*, *2*(3), 64–72.

Punuri, S. B., Kuanar, S. K., Kolhar, M., Mishra, T. K., Alameen, A., Mohapatra, H., & Mishra, S. R. (2023). Efficient Net-XGBoost: An Implementation for Facial Emotion Recognition Using Transfer Learning. *Mathematics*, *11*(3). https://doi.org/10.3390/math11030776

Ridhovan, A., & Suharso, A. (2022). Penerapan Metode Residual Network (Resnet) Dalam Klasifikasi Penyakit Pada Daun Gandum. *JIPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika)*, *7*(1), 58–65. https://doi.org/10.29100/jipi.v7i1.2410

Rozaqi, A. J., Sunyoto, A., & Arief, M. rudyanto. (2021). Deteksi Penyakit Pada Daun Kentang Menggunakan Pengolahan Citra dengan Metode Convolutional Neural Network. *Creative Information Technology Journal*, *8*(1), 22. https://doi.org/10.24076/citec.2021v8i1.263

Tan, M., & Le, Q. (2019). {E}fficient{N}et: Rethinking Model Scaling for Convolutional Neural Networks. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th International Conference on Machine Learning* (Vol. 97, pp. 6105–6114). PMLR. https://proceedings.mlr.press/v97/tan19a.html

Tangkelayuk, A. (2022). The Klasifikasi Kualitas Air Menggunakan Metode KNN, Naïve Bayes, dan Decision Tree. *JATISI (Jurnal Teknik Informatika Dan Sistem Informasi)*, *9*(2), 1109–1119. https://doi.org/10.35957/jatisi.v9i2.2048

Veneman, R. (2021). *Real-time skin cancer detection using neural networks on an embedded device*.

Windiawan, R., Suharso, A., & Artikel, S. (2019). Identifikasi Penyakit pada Daun Kopi Menggunakan Metode Deep Learning VGG16 INFO ARTIKEL ABSTRAK. *Exploreit*, *13*(2), 9–16. https://doi.org/10.35891/explorit

Yandika, R. F., Irawati, A. R., Komputer, J. I., Lampung, U., & Lampung, B. (2023). Pengembangan Aplikasi Kelas Ibu Hamil Berbasis Android Menggunakan Framework. *Kumpulan JurnaL Ilmu Komputer (KLIK)*, *10*, 320–331.

Yang, X., Zhang, S., Liu, J., Gao, Q., Dong, S., & Zhou, C. (2021). Deep learning for smart fish farming: applications, opportunities and challenges. *Reviews in Aquaculture*, *13*(1), 66–90. https://doi.org/https://doi.org/10.1111/raq.12464

Zuhan, M., & Kristian, Y. (2023). Detection of Porang Plant Diseases and Pests (Amorphophallus Muelleri) Based on Leaf Imagery Utilizing DCNN Transfer Learning. *JTECS : Jurnal Sistem Telekomunikasi Elektronika Sistem Kontrol Power Sistem Dan Komputer*, *3*(2), 129. https://doi.org/10.32503/jtecs.v3i2.3709