

**PENGEMBANGAN SISTEM DETEKSI PENYAKIT DAUN TANAMAN
TOMAT MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL
NETWORK* PADA SISTEM OPERASI ANDROID**

SKRIPSI

Untuk memenuhi salah satu syarat mencapai derajat pendidikan Strata Satu (S-1)
Sebagai Sarjana Sains pada Departemen Fisika



Disusun Oleh:

**Maura Tsaabitah Suci Prayitno
24040120140142**

**PROGRAM STUDI FISIKA
DEPARTEMEN FISIKA
FAKULTAS SAINS DAN MATEMATIKA
UNIVERSITAS DIPONEGORO
SEMARANG
Agustus, 2024**

PERSETUJUAN UJIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini Dosen Pembimbing dari:

Mahasiswa : Maura Tsaabitah Suci Prayitno
NIM : 24040120140142
Jurusan/Fakultas : Fisika/FSM
Judul Skripsi : Pengembangan Sistem Deteksi Penyakit Daun
Tanaman Tomat Menggunakan Metode
Convolutional Neural Network Pada Sistem Operasi
Android

Menyatakan bahwa mahasiswa tersebut telah melaksanakan Ujian Seminar Hasil Skripsi sehingga menyetujui dan layak untuk melaksanakan Ujian Tugas Akhir.

Semarang, 22 Agustus 2024

Dosen Pembimbing I,

Dosen Pembimbing II,

(Prof. Dr. Kusworo Adi, S. Si., M. T.)
NIP. 197203171998021001

(Dr. Drs. Catur Edi Widodo, M. T.)
NIP. 196405181992031002

PERNYATAAN ORISINALITAS

Dengan ini saya menyatakan bahwa dalam skripsi ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar akademis di suatu perguruan tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Semarang, 22 Agustus 2024

Materai 10.000

Maura Tsaabitah Suci Prayitno
NIM. 24040120140142

**HALAMAN PENGESAHAN
SKRIPSI**

**Pengembangan Sistem Deteksi Penyakit Daun Tanaman Tomat Menggunakan
Metode *Convolutional Neural Network* Pada Sistem Operasi Android**

**Disusun Oleh:
Maura Tsaabitah Suci
24040120140142**

**Telah diujikan dan dinyatakan lulus oleh Tim Penguji
Pada tanggal Agustus 2023**

**Tim penguji,
Dosen Pembimbing I , Dosen Pembimbing II,**

**(Prof. Dr. Kusworo Adi, S. Si., M. T.) (Dr. Drs. Catur Edi Widodo, M. T)
NIP. 197203171998021001 NIP. 196405181992031002**

Penguji I, Penguji II,

**(Dr. Dra. Sumariyah, M.Si.) (Heri Sugito, S.Si., M.Si., F.Med.)
NIP. 196103101988032001 NIP. 198010072005011002**

Penguji III,

**(Zaenal Arifin, S.Si., M.Si., F.Med.)
NIP. 197705102008121001**

**Skripsi ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Sarjana Sains (S.Si)**

**Tanggal, Agustus 2023
Ketua Departemen Fisika,**

**Prof. Dr. Heri Sutanto, S.Si., M.Si., F.Med
NIP. 197502151998021001**

**PERNYATAAN PERSETUJUAN
PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS**

Sebagai civitas akademik Universitas Diponegoro, saya yang bertandatangan di bawah ini:

Mahasiswa	: Maura Tsaabitah Suci Prayitno
NIM	: 24040120140142
Program Studi	: Fisika
Jurusan	: Fisika
Fakultas	: Sains dan Matematika
Jenis Karya	: Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Hak Bebas Royalti Noneksklusif atas karya ilmiah saya yang berjudul:

**PENGEMBANGAN SISTEM DETEKSI PENYAKIT TANAMAN TOMAT
MENGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK*
(CNN) PADA SISTEM OPERASI ANDROID**

Beserta perangkat yang ada. Dengan Hak Bebas Royalti Noneksklusif ini Program Studi Fisika Fakultas Sains dan Matematika Universitas Diponegoro berhak menyimpan, mengalih media/ formatkan, mengelola dalam bentuk pangkalan (database) merawat, dan mempublikasikan skripsi saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik hak cipta.

Dibuat di : Semarang

Pada Tanggal : 22 Agustus 2024

Yang menyatakan

Meterai 10.000

Maura Tsaabitah Suci Prayitno
24040120140142

KATA PENGANTAR

Puji syukur penulis ucapkan ke hadirat Allah SWT yang telah memberikan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir ini yang berjudul “Pengembangan Sistem Deteksi Penyakit Tanaman Tomat Menggunakan Metode Convolutional Neural Network (CNN) Pada Sistem Operasi Android”.

Dalam penulisan skripsi ini, banyak pihak-pihak yang telah membantu hingga karya tulis ini dapat penulis selesaikan. Oleh karena itu, penulis berterima kasih kepada :

1. Mamah yuli, Abi Teguh, dan juga segenap keluarga penulis yang selalu memberi motivasi, dukungan, dan doa demi kelancaran proses pengerjaan tugas akhir ini.
2. Prof. Dr. Kusworo Adi, S. Si., M. T. selaku dosen pembimbing I yang telah mengarahkan dalam proses pembuatan Tugas Akhir.
3. Dr. Drs. Catur Edi Widodo, M. T. selaku dosen pembimbing II yang telah memberi pencerahan bidang ilmu terkait Tugas Akhir.
4. Ibu Dr. Dra. Sumariyah, M.Si., Bapak Heri Sugito, S.Si., M.Si., F.Med., dan Bapak Zaenal Arifin, S.Si., M.Si., F.Med., selaku dosen penguji pada sidang skripsi ini.
5. Afifah Auliawati dan Shabrina Sekarayu Wiboyo selaku sahabat penulis yang telah menemani penulis selama perkuliahan.
6. Shafanisa Alifia selaku sahabat penulis yang telah menemani, mendukung dan menyemangati penulis selama 10 tahun terakhir.
7. Dania, Khansa, Nandini, Alma, Anet, Najla, Zahra selaku sahabat penulis yang selalu mendukung dan menyemangati penulis selama masa perkuliahan.
8. Hans Filbert Lee selaku teman satu bimbingan dan juga orang terdekat penulis yang telah menemani, menyemangati dan membantu penulis untuk banyak hal.
9. Kaggle yang menyediakan dataset yang sangat berguna bagi penelitian ini.

Semoga tugas akhir ini dapat bermanfaat, baik sebagai sumber informasi maupun sumber inspirasi bagi para pembaca.

Semarang, 22 Agustus 2024

Maura Tsaabitah Suci Prayitno

DAFTAR ISI

Halaman Judul.....	ii
Persetujuan Ujian Tugas Akhir	ii
Pernyataan Orisinalitas.....	iii
Halaman Pengesahan	iv
Pernyataan Persetujuan Publikasi Skripsi Untuk Kepentingan Akademis	v
Kata Pengantar	vi
Daftar Isi	viii
Daftar Tabel	x
Daftar Gambar.....	xi
Daftar Lampiran	xii
Abstrak	xiv
<i>Abstract</i>	xv
Bab I Pendahuluan	1
1.1 Latar Belakang.....	1
1.2 Tujuan Penelitian.....	3
1.3 Manfaat Penelitian.....	3
Bab II Dasar Teori.....	4
2.1 Penyakit Tomat.....	4
2.1.1 <i>Mosaic Virus</i>	4
2.1.2 <i>Target Spot</i>	5
2.1.3 <i>Bacterial Spot</i>	6
2.1.4 <i>Yellow Leaf Curl Virus</i>	6
2.1.5 <i>Late Blight</i>	7
2.1.6 <i>Leaf Mold</i>	8
2.1.7 <i>Early Blight</i>	8
2.1.8 <i>Spider Mites</i>	9
2.1.9 <i>Septoria Leaf Spot</i>	10
2.2 Pengolahan Citra Digital	10
2.3 <i>Convolutional Neural Network (CNN)</i>	13
2.4 Python.....	14
2.5 TensorFlow	15

2.6	Android Studio	16
Bab III Metode Penelitian		18
3.1	Tempat dan Waktu Penelitian	18
3.2	Alat dan Bahan Penelitian	18
3.3	Prosedur Penelitian.....	19
3.4	Diagram Alir.....	21
Bab IV Hasil Dan Pembahasan		22
4.1	Pengolahan Data.....	22
4.2	Hasil Pelatihan Model	22
4.1.1	Pelatihan Model dengan 3 Lapisan Konvolusi Tanpa <i>Reduce Learning Rate</i> dan <i>Early Stopping</i>	23
4.1.2	Pelatihan Model dengan Penambahan Lapisan Konvolusi dan Regularisasi	25
4.1.3	Pelatihan Model dengan Penambahan <i>Reduce Learning Rate</i>	28
4.3	Pembahasan	29
4.4	Pengujian	30
4.5	Implementasi Model Dalam Aplikasi Android	33
Bab V Kesimpulan		35
5.1	Kesimpulan.....	35
5.2	Saran	35
Daftar Pustaka		36
Lampiran		39

DAFTAR TABEL

Tabel 3.1 Dataset Citra Daun Tomat.....	19
Tabel 4.1 Model Pelatihan Pertama	25
Tabel 4.2 Model CNN yang Telah Diperbarui	27
Tabel 4.3 Citra yang Gagal Diidentifikasi	32

DAFTAR GAMBAR

Gambar 2.1 <i>Mosaic Virus</i>	5
Gambar 2.2 <i>Target Spot</i>	5
Gambar 2.3 <i>Bacterial Spot</i>	6
Gambar 2.4 <i>Yellow Curl Leaf Virus</i>	7
Gambar 2.5 <i>Late Blight</i>	7
Gambar 2.6 <i>Leaf Mold</i>	8
Gambar 2.7 <i>Early Blight</i>	9
Gambar 2.8 <i>Spider Mites</i>	9
Gambar 2.9 <i>Septoria Leaf Spot</i>	10
Gambar 2.10 Matrix Citra RGB	11
Gambar 2.11 Matrix Citra <i>Greyscale</i>	11
Gambar 2.12 (a) Citra <i>greyscale</i> dari Karakter 'A' (b) Representasi Biner dari Karakter 'A' (c) Representasi Matriks Biner	12
Gambar 2.13 Arsitektur CNN	13
Gambar 2.14 Arsitektur TensorFlow	16
Gambar 3.1 Diagram Alir Penelitian	20
Gambar 3.2 Diagram Alir	22
Gambar 4.1 Dataset Citra Tanaman Tomat	23
Gambar 4.2 Grafik hasil pelatihan dengan 3 lapisan konvolusi	26
Gambar 4.3 Grafik hasil pelatihan dengan penambahan lapisan konvolusi dan Regularisasi.....	28
Gambar 4.4 Grafik hasil pelatihan dengan penambahan <i>reduce learning rate</i>	29
Gambar 4.5 <i>Confusion Matrix</i> Data Uji	31
Gambar 4.6 Tampilan UI Aplikasi Pada Sistem Operasi Android	34

DAFTAR LAMPIRAN

Lampiran 1 Coding pelatihan model dengan 3 lapisan konvolusi tanpa <i>Reduce Learning Rate</i> dan <i>Early Stopping</i>	40
Lampiran 2 Coding Pelatihan Model dengan Penambahan Lapisan Konvolusi dan Regularisasi	43
Lampiran 3 Coding Pelatihan Model Dengan Penambahan <i>Reduce Learning Rate</i>	46
Lampiran 4 Coding Testing Model Terbaik	50
Lampiran 5 Coding User Interface Aplikasi	53
Lampiran 6 Coding Kotlin Untuk Fitur Aplikasi	55
Lampiran 7 Tampilan Aplikasi Pada Sistem Operasi Android.....	58

ARTI LAMBANG DAN SINGKATAN

CNN	: <i>Convolutional Neural Network</i>
ToMV	: <i>Tomato Mosaic Virus</i>
RGB	: <i>Red, Green, Blue</i>
IDE	: <i>Integrated Development Environment</i>
2D	: <i>2 Dimention</i>
TfLite	: <i>TensorFlow Lite</i>
UI	: <i>User Interface</i>
Tc	: Total sampe citra teridentifikasi benar
E	: error atau jumlah kesalahan dalam identifikasi

ABSTRAK

Penurunan produksi tomat di Indonesia yang mencapai 12.3% pada desember 2023 disebabkan oleh perubahan musim dan serangan hama dan penyakit. Untuk mengatasi masalah ini diperlukan aplikasi yang dapat membantu menemukan serta mengenali penyakit pada tanaman tomat. Pada penelitian ini aplikasi dioperasikan pada sistem operasi android dengan sistem deteksi menggunakan metode *deep learning* yaitu *convolutional neural network* (CNN). Dataset yang digunakan pada penelitian ini berjumlah 5000 citra daun tomat yang diperoleh dari Kaggle. Citra tersebut melalui tahap preprocessing kemudian dibagi menjadi data latih 80%, validasi 10% dan uji 10%. Data kemudian dilatih menggunakan model dengan 3 variasi parameter. Dari ketiga variasi tersebut diperoleh hasil akurasi terbaik sebesar 0.9995 pada data pelatihan dan 0.9833 pada data validasi, dengan nilai *loss* sebesar 0.0759 digunakan sebagai data pelatihan dan 0.1099 digunakan sebagai data validasi. Model dengan akurasi terbaik kemudian dikonversi ke TensorFlow Lite dan diintegrasikan ke dalam aplikasi Android. Pengujian sistem deteksi dengan 520 data uji menunjukkan akurasi keseluruhan 98.8%. Hasil ini menunjukkan bahwa sistem deteksi yang dikembangkan mampu mendeteksi penyakit pada daun tanaman tomat dengan akurasi tinggi, sehingga dapat membantu petani dalam mengidentifikasi penyakit secara dini dan meningkatkan produktivitas tanaman tomat.

Kata Kunci : Tomat, Penyakit Tanaman, *Convolutional Neural Network*, *Android*

ABSTRACT

The decrease in tomato production in Indonesia, reaching 12.3% in December 2023, is caused by seasonal changes, pest and disease attacks. To address this issue, an application is needed to help detect and recognize diseases in tomato plants. In this study, an application was developed for the Android operating system, utilizing deep learning methods, specifically Convolutional Neural Networks (CNN). The dataset used in this research consisted of 5,000 images of tomato leaves obtained from Kaggle. These images underwent preprocessing and were then divided into training data (80%), validation data (10%), and test data (10%). The data were then trained using a model with three parameter variations. Among these variations, the best accuracy was achieved at 0.9995 for the training data and 0.9833 for the validation data, with a loss value of 0.0759 for the training data and 0.1099 for the validation data. The model with the best accuracy was then converted to TensorFlow Lite and integrated into an Android application. System testing with 520 test data showed an overall accuracy of 98.8%. These results indicate that the developed detection system is capable of identifying diseases in tomato plant leaves with high accuracy, thereby assisting farmers in early disease identification and improving tomato plant productivity.

Keywords : *Tomato, Plant Disease, Convolutional Neural Network, Android*

BAB I

PENDAHULUAN

1.1 Latar Belakang

Solanum Lycopersicum juga dikenal dengan tomat termasuk kedalam buah yang cukup diminati oleh masyarakat. Buah ini seringkali di konsumsi dengan atau tanpa pengolahan terlebih dahulu. Tomat termasuk dalam jenis buah yang kaya akan kandungan Vitamin A dan C yang cukup tinggi. Tomat juga menjadi salah satu asal produksi likopen terbaik, dimana likopen dapat diolah menjadi produk kesehatan yang bernilai tinggi (Hadi, 2023). Karena kandungan akan vitamin dan nutrisi yang tinggi menjadikan permintaan konsumen terhadap buah tomat cukup tinggi. Namun di Indonesia proyeksi produksi tomat mengalami penurunan. Hal ini dapat dilihat dari perkembangan data produksi dan harga di tingkat konsumen bahwa penurunan produksi sebesar 12,3% pada bulan Desember 2023 mempengaruhi kenaikan harga (Purmadani, 2024). Fluktuasi produksi tomat dapat disebabkan oleh beberapa faktor seperti perubahan musim, ukuran lahan pertanian, sistem pertanian yang digunakan, serta serangan hama dan penyakit tanaman.

Namun, serangan hama dan penyakit seringkali menjadi faktor dominan yang menyebabkan produksi tomat berkurang (Baideng, 2016). Untuk mengatasi hal tersebut diperlukan peningkatan produksi tomat. Salah satu Upaya untuk meningkatkan kuantitas produksi adalah dengan pengendalian hama dan penyakit. Tanaman tomat memiliki banyak jenis penyakit yang dapat terdeteksi dari daunnya. Namun, para petani sering menghadapi kesulitan dalam mengidentifikasi secara tepat jenis penyakit yang mungkin dialami oleh tanaman. Keterlambatan dalam mengidentifikasi penyakit ini dapat mengakibatkan penyakit tersebut menyebar ke seluruh lahan dan mengakibatkan kerugian yang besar (Chaudhari, dkk., 2019). Dengan demikian, upaya lebih lanjut untuk meningkatkan kemampuan identifikasi penyakit tanaman tomat sangat diperlukan untuk meminimalisir dampak negatifnya.

Berdasarkan masalah yang telah dijelaskan, sistem deteksi penyakit tanaman tomat merupakan solusi yang sangat efektif untuk mengurangi persentase gagal panen yang kerap dialami oleh petani. Dengan adanya sistem deteksi ini, para petani akan lebih mudah dalam mengidentifikasi penyakit yang mungkin menyerang tanaman mereka secara *real-time*. Selain itu, pengembangan sistem ini perlu dilakukan agar kedepannya pasokan tanaman tomat dapat terus terpenuhi, sehingga memberikan kontribusi yang signifikan dalam meningkatkan kesejahteraan para petani. Pengembangan sistem deteksi penyakit tanaman tomat dapat dilakukan dengan memanfaatkan kemajuan teknologi pada zaman sekarang.

Teknik – Teknik visi komputer berbasis *deep learning* memberikan cara baru untuk klasifikasi citra, deteksi objek, dan sebagainya. *Deep learning* merupakan *subfield* dari *Machine Learning* yang berfokus pada pengembangan dan pelatihan *artificial neural networks* dengan beberapa lapisan. *Deep Learning* dirancang untuk membuat komputer mempelajari dan membuat Keputusan atau prediksi tanpa diprogram secara eksplisit (Shah, 2024). *Convolutional Neural Network* (CNN) adalah salah satu arsitektur *Deep Learning* yang sangat terkenal untuk visi komputer. Metode ini dapat mempelajari fitur-fitur dari setiap lapisan tersembunyi dari data input secara otomatis. Metode ini telah banyak digunakan dalam penelitian pengolahan citra. Samer I. Mohamed (2020) melakukan penelitian yang menggunakan CNN untuk menganalisis citra daun kentang. Penelitian tersebut menghasilkan akurasi yang tinggi sebesar 98%. Akan tetapi penelitian tersebut tidak diimplementasikan untuk penggunaan *real-time*.

Telepon genggam dengan sistem operasi android merupakan salah satu piranti yang paling umum digunakan di Indonesia karena kemudahan akses yang ditawarkannya. Android merupakan sistem operasi berbasis linux. Para pengembang diberikan kerangka kerja untuk yang lengkap dan beragam untuk mengembangkan berbagai jenis aplikasi yang dapat diakses oleh berbagai jenis perangkat dengan sistem operasi android (Murya, 2014).

1.2 Tujuan Penelitian

Penelitian ini bertujuan untuk membangun dan mengimplementasikan sistem deteksi pada sistem operasi android yang dapat mendeteksi penyakit pada daun tanaman tomat.

1.3 Manfaat Penelitian

Adapun manfaat dari penelitian ini, yaitu:

1. Memudahkan para petani atau penanam tomat awam dalam mendeteksi penyakit yang dialami oleh tanaman tomat agar dapat melakukan penanganan dini dan mengurangi resiko gagal panen.
2. Membantu petani dalam meningkatkan produktivitas tanaman dan hasil panen secara keseluruhan melalui upaya deteksi dini penyakit pada tanaman tomat.

BAB II

DASAR TEORI

2.1 Penyakit Tomat

Penyakit tanaman tomat merupakan permasalahan yang cukup serius bagi para petani. Tanaman yang terkena penyakit dapat mempengaruhi produktivitas dan kualitas tanaman yang menyebabkan kerugian hasil panen. Penyakit tomat dapat disebabkan oleh berbagai hal seperti musim, cuaca, virus, jamur dan juga hama. Jenis penyakit tanaman tomat yang berbeda memiliki cara penanganan yang berbeda pula, waktu penanganan juga sangat berpengaruh terhadap hasil panen. Oleh karena itu, deteksi dini tanaman penyakit tomat penting untuk mengurangi resiko gagal panen (Liu & Wang, 2020).

2.1.1 Mosaic Virus

Genus *Tobamavirus* memiliki beberapa jenis salah satunya adalah *Tomato mosaic virus* (ToMV). *Tobamavirus* dikenal karena keunikannya yang memiliki kemampuan untuk menyebar ke tanaman inang tanpa bantuan vektor hidup. *Tobamavirus* juga dikenal sangat tangguh dan tetap menular di tanah dan sisa-sisa tanaman yang terkontaminasi, air irigasi yang diambil dari sungai, dan juga dalam larutan nutrisi dalam sistem budaya hidroponik dalam waktu yang lama. Diantara virus lainnya, *Tomato mosaic virus* sangat berbahaya, dengan kisaran inang yang luas sehingga menyebabkan gejala mosaic pada tanaman inangnya (Mrkvová, dkk., 2022). Gejala ini meliputi perubahan warna dan pola pada daun seperti pada Gambar 2.1. Virus ini sangat berdampak pada hasil dan kualitas panen yang dihasilkan para petani.



Gambar 2.1 *Mosaic Virus* (Wati, dkk., 2021)

2.1.2 Target Spot

Penyakit tanaman tomat yang disebabkan oleh jamur *Corynespora cassiicola* adalah penyakit *target spot*. Gejala utama pada penyakit *target spot* meliputi bintik-bintik coklat yang tidak beraturan pada daun, sering kali diapit oleh lingkaran hijau kekuningan yang mencolok seperti pada Gambar 2.2. Fenomena pertumbuhan bintik-bintik ini seringkali menghasilkan pola sebaran berupa cincin-cincin coklat terang atau gelap yang terlihat menyerupai "target", sehingga penyakit ini dikenal dengan sebutan *target spot*. Namun, tidak hanya daun yang terpengaruh, batang dan tangkai pun dapat menunjukkan gejala serupa, ditandai dengan bintik-bintik coklat gelap atau luka-luka memanjang. Infeksi yang parah dapat menyebabkan daun rontok sebelum waktunya (Kamei, dkk., 2018)



Gambar 2.2 *Target Spot* (Kamei, dkk., 2018)

2.1.3 Bacterial Spot

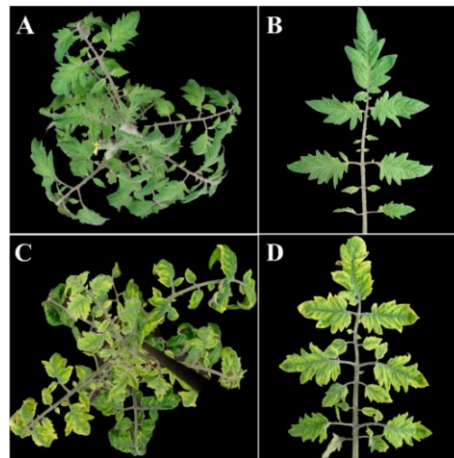
Bacterial spot pada tanaman tomat adalah penyakit yang berpotensi merusak bahkan dalam kasus yang parah dapat menyebabkan kematian tanaman dan menyebabkan buah tidak dapat dijual. Penyakit *bacterial spot* pada tanaman tomat dapat disebabkan oleh bakteri *Xanthomonas citri* pv. *Mangiferaeindicae*. *Bacterial spot* dapat terjadi di mana pun tanaman tomat tumbuh, tetapi paling sering ditemukan di iklim yang hangat dan lembab. Gejala penyakit *Bacterial Spot* berbentuk bulat kecil yang dapat ditemukan pada daunnya seperti pada Gambar 2.3. Bercak pada awalnya berwarna kuning kehijauan, tetapi menjadi lebih gelap menjadi coklat kemerahan seiring berjalannya waktu (Marks, 2017).



Gambar 2.3 *Bacterial Spot* (Sitthitanasin, dkk., 2021)

2.1.4 Yellow Leaf Curl Virus

Penyakit yang dapat mengganggu produksi tomat lainnya adalah penyakit keriput daun kuning tomat. Penyakit ini disebabkan oleh virus tumbuhan yang ditularkan oleh kutu putih (Li, dkk., 2021). Virus Keriput Daun Kuning Tomat adalah *begomovirus monopartit* dalam famili *Geminiviridae* dan merupakan salah satu dari banyak virus yang menyebabkan penyakit keriput daun kuning pada tomat (Mabvakure, dkk., 2016). Perubahan warna daun menjadi kekuningan dan keriput di bagian pinggiran daun dan dapat menyebar merupakan penanda bahwa daun tersebut terkena Virus Keriput Daun Kuning Tomat. Gejala tersebut dapat dilihat pada Gambar 2.4 dimana Gambar A dan B adalah daun yang sehat dan Gambar C dan D adalah daun yang terinfeksi.



Gambar 2.4 *Yellow Curl Leaf Virus* (Yan, dkk., 2021)

2.1.5 Late Blight

Late Blight atau penyakit busuk pada tanaman tomat adalah penyakit tanaman disebabkan oleh jamur *Phytophthora infestans* yang menyerang tanaman tomat. Tingkat kerusakan tanaman akibat penyakit *late blight* mencapai 100% tergantung cuaca dan kultur teknis yang tentunya dapat menurunkan hasil panen (Wiguna, dkk., 2015). Gejala tanaman tomat yang telah diserang oleh jamur *Phytophthora infestans* dapat terlihat pada daunnya yang berubah warna menjadi coklat kehitaman karena terjadi nekrosis seperti pada Gambar 2.5. Hal ini dapat terus menyebar ke seluruh daun tanaman baik yang muda maupun tua jika tidak segera ditangani jamur dapat menyebar ke batang dan buah tanaman yang akan berdampak buruk pada hasil panen.



Gambar 2.5 *Late Blight* (Wati, dkk., 2021)

2.1.6 Leaf Mold

Penyakit yang dikenal sebagai "*leaf mold*" pada tanaman tomat memiliki beberapa gejala yang dapat diidentifikasi. Pertama, terlihat adanya lesi berbentuk bola atau *elips* berwarna hijau pada sisi atas daun yang lebih tua. Kemudian, proses nekrosis atau kematian jaringan daun dapat terjadi, terutama setelah infeksi berkembang menjadi lebih besar. Pemecahan atau kerusakan daun juga dapat muncul, terutama ketika lesi mencapai ukuran yang cukup besar dan tanaman menjadi tidak mampu menopang berat daun. Semua gejala ini disebabkan oleh jamur *Fulvia fulva* (Cooke) atau *Cladosporium fulvum* (Latorre & Besoain, 2007). Gambar 2.6 menunjukkan daun yang terkena penyakit leaf mold.



Gambar 2.6 Leaf Mold (Wagle, dkk., 2022)

2.1.7 Early Blight

Early blight adalah penyakit yang mempengaruhi tanaman tomat dan disebabkan oleh jamur *Alternaria solani*. *Early Blight* merupakan penyakit tanaman tomat yang bisa mengganggu. Akan tetapi, jika tingkat infeksi rendah pada daun-daun bagian bawah, tidak akan mempengaruhi hasil panennya karena setelah tanaman mulai berbuah, daun-daun bagian bawah akan mengalami penuaan dan gugur secara alami. Namun jika Tingkat infeksi cukup tinggi maka perlu dilakukan pengendalian penyakit secepatnya agar tidak menyebar ke daun yang lebih muda (Watt, 2020). Daun tomat yang terjangkit *early blight* dapat dilihat pada Gambar 2.7.



Gambar 2.7 *Early Blight* (Wati, dkk., 2021)

Gejala awal penyakit ini dapat dilihat dengan munculnya bintik kecoklatan berbentuk lingkaran yang memiliki detail seperti cincin lalu membesar dan menutupi seluruh daun sehingga membuat daun gugur.

2.1.8 Spider Mites

Spider Mites atau kutu laba-laba adalah hama yang sering mengganggu tanaman tomat. Kutu laba-laba berbintik dua (*Tetranychus urticae*) spesies tertentu yang dapat menyebabkan kerusakan pada tanaman tomat. Gejala tanaman tomat yang terkena kutu laba-laba ini dapat di tandai dengan adanya jaring-jaring halus pada bagian bawah daun, munculnya bercak kecil pada daun yang disebut *stippling* dan perubahan warna daun menjadi kuning atau coklat seperti yang ditunjukkan pada Gambar 2.8 (Acosta & Cañas, 2019).



Gambar 2.8 *Spider Mites* (Özb' ilge, dkk., 2022)

2.1.9 Septoria Leaf Spot

Septoria leaf spot adalah penyakit yang mempengaruhi tanaman tomat dan disebabkan oleh jamur *Septoria lycopersici*. Penyakit ini dapat menyerang daun tanaman tomat yang masih muda ataupun yang sudah tua. Gejala ringan penyakit *Septoria leaf spot* pada tanaman tomat ditandai dengan adanya bercak kecil berwarna coklat hingga hitam yang dapat ditemukan pada sisi permukaan daun. Jika tidak ditangani penyakit ini dapat memperburuk bercak kecil ini hingga menutupi seluruh daun seperti pada Gambar 2.9. Jika bercak sudah menyebar ke seluruh daun maka akan menyebabkan kerontokan pada daun (Sindushree, dkk., 2020).



Gambar 2.9 *Septoria Leaf Spot* (Wati, dkk., 2021)

2.2 Pengolahan Citra Digital

Citra digital merupakan gambar analog dua dimensi yang direpresentasikan, yang kemudian diubah menjadi gambar diskrit melalui proses *sampling*. Proses *sampling* ini membagi gambar analog menjadi N baris dan M kolom, menghasilkan citra digital yang terdiri dari kumpulan nilai diskrit. Citra digital merupakan bentuk gambar yang dapat diproses oleh komputer, di mana informasi gambar disimpan dalam bentuk angka-angka yang mewakili besar intensitas pada setiap piksel. Karena bersifat data numerik, citra digital dapat diolah dan dianalisis menggunakan komputer (Muzahardin, dkk., 2022).

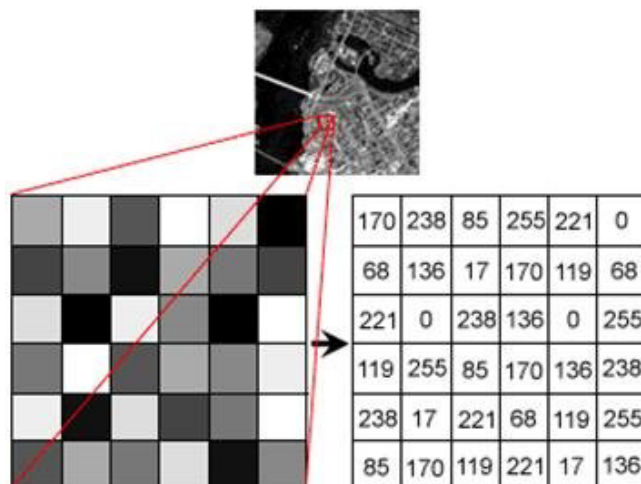
Berdasarkan nilai pikselnya citra digital dapat dikelompokkan ke dalam 3 jenis yaitu Citra RGB, Citra *Grayscale* dan Citra Biner. Citra RGB yaitu jenis citra digital yang menggunakan model warna RGB untuk menggambarkan nilai warna pada setiap piksel. Model warna RGB menggunakan tiga komponen warna yaitu

merah (R), hijau (G), dan biru (B), yang setiap komponen memiliki nilai antara 0 dan 255. Citra ini digunakan untuk representasi dan tampilan gambar pada sistem elektronik seperti televisi, komputer, dan masih banyak lagi. Gambar 2.10 adalah gambar representasi matriks citra RGB.

					165	187	209	58	7
					14	125	233	201	98
253	144	120	251	41	147	204			
67	100	32	241	23	165	30			
209	118	124	27	59	201	79			
210	236	105	169	19	218	156			
35	178	199	197	4	14	218			
115	104	34	111	19	196				
32	69	231	203	74					

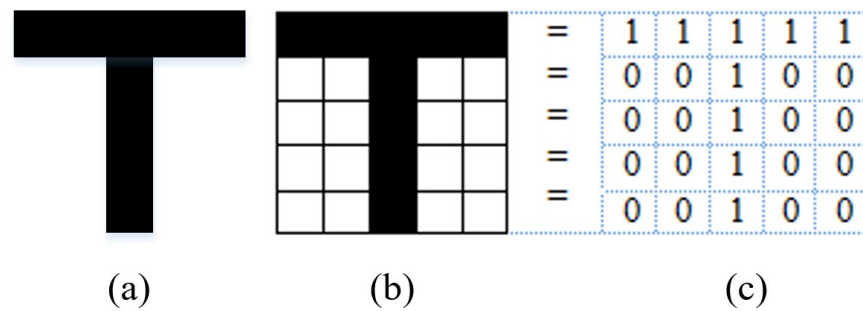
Gambar 2.10 Matrix Citra RGB (Courtney, 2001)

Terdapat citra yang menggunakan skala warna abu-abu yang tersusun dari gradasi antara hitam dan putih biasanya disebut sebagai citra berskala keabuan (*grayscale*). Setiap tingkat keabuan pada citra *grayscale* direpresentasikan dengan nilai bilangan bulat mulai dari 0 hingga 255, di mana 0 mewakili warna hitam dan 255 mewakili warna putih. Nilai tersebut mencerminkan Tingkat intensitas dari warna abu-abu yang bersangkutan seperti yang ditunjukkan oleh Gambar 2.11.



Gambar 2.11 Matrix Citra *Greyscale* (Neves, dkk., 2018)

Citra biner adalah jenis citra digital yang hanya terdiri dari dua warna yaitu hitam dan putih. Setiap piksel dalam citra biner hanya memiliki nilai 0 atau 1, di mana nilai 0 merepresentasikan warna putih dan 1 warna hitam seperti pada Gambar 2.12. (Majid, dkk., 2022)



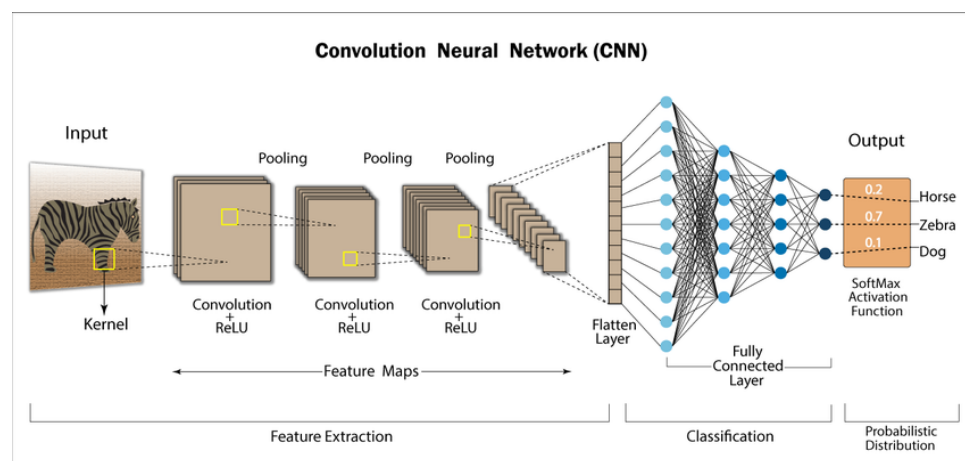
Gambar 2.12 (a) Citra *grayscale* dari Karakter 'T' (b) Representasi Biner dari Karakter 'A' (c) Representasi Matriks Biner (Choudhary, dkk., 2010)

Ilmu dalam memodifikasi citra digital dengan bantuan komputer disebut dengan pengolahan citra digital (Indra, dkk., 20118). Proses modifikasi dilakukan dengan tujuan meningkatkan kualitas citra sehingga manusia dan komputer dapat lebih mudah memahaminya. Terdapat tahap input dan output dalam proses pengolahan citra . Tahap input yaitu tahap dimana citra asli dimasukkan ke dalam sistem komputer untuk diproses lebih lanjut sedangkan tahap output adalah tahap ketika citra telah selesai diproses dan disajikan dalam bentuk yang diinginkan seperti gambar (Devi & Rosyid, 2022).

Preprocessing citra adalah proses mengolah citra sebelum dilakukan pengolahan lanjutan, seperti melakukan peningkatan kontras, menyamakan ukuran gambar, mengurangi *noise* pada gambar untuk memudahkan proses pengolahan citra yang akan dilakukan selanjutnya. Perbaikan kualitas citra atau *Image Enhancement* merupakan salah satu tahapan penting dalam *Preprocessing*. Pada tahap ini, dilakukan perbaikan kontras gelap atau terang dan penajaman citra sehingga ekstraksi fitur citra dapat dilakukan dengan baik. Selain itu, *image restoration* juga dapat dilakukan pada saat *Preprocessing*, yaitu dengan menghilangkan *noise* pada citra (Sari, dkk., 2020).

2.3 Convolutional Neural Network

Convolutional Neural Network (CNN) adalah arsitektur *deep learning* yang sering digunakan untuk mengatasi masalah klasifikasi gambar. CNN adalah jenis *neural network* yang diterapkan dalam pengolahan citra digital, yang memiliki arsitektur yang berbeda dengan jenis *neural network* lainnya. Salah satu contoh arsitektur CNN dapat dilihat pada Gambar 2.13. CNN menggunakan metode konvolusi, yang berfungsi untuk mengumpulkan informasi dari citra dan mengurangi jumlah parameter yang dibutuhkan (Azmi, dkk., 2023).



Gambar 2.13 Arsitektur CNN (Kumar, dkk., 2023)

Metode *Convolutional Neural Network* memiliki popularitas tinggi pada kalangan *deep learning*, dikarenakan CNN mengekstrak fitur dari input yang berupa gambar kemudian mengurangi dimensi gambar tersebut menjadi lebih kecil tanpa mengubah karakteristiknya (Omori & Shima, 2020). *Convolutional Neural Network* (CNN) terdiri dari *neurons* yang memiliki bobot dan bias, *neurons* tersebut menerima inputan dan dilanjutkan dengan melakukan perkalian titik pada setiap *neuron* tersebut. CNN terdiri dari tiga lapisan utama yaitu *Convolutional Layer*, *Pooling Layer*, dan *Fully Connected Layer* (Ersyad, dkk., 2020).

Convolutional layer merupakan bagian penting dari *neural network* untuk memproses gambar. Lapisan ini menggunakan filter untuk mengenali pola dan fitur dalam gambar. Dengan mengurangi jumlah parameter yang dibutuhkan, lapisan ini membantu jaringan untuk belajar fitur-fitur yang penting dari gambar. Bobot pada

lapisan ini menentukan filter yang digunakan untuk mengenali pola dalam gambar. Dengan berbagi parameter antara piksel-piksel yang berdekatan, *convolutional layer* memungkinkan jaringan untuk memahami gambar secara lebih efisien (Alwanda, dkk., 2020).

Pooling layer adalah komponen penting dari jaringan *neural* yang digunakan untuk meningkatkan ketahanan jaringan terhadap *overfitting* dan mengurangi jumlah parameter yang diperlukan. Pada CNN, *pooling layer* biasanya digunakan setelah *convolutional layer* untuk mengurangi dimensi gambar dan meningkatkan ketahanan jaringan. Jenis *pooling layer* yang paling umum digunakan adalah *max pooling*, yang memilih nilai maksimum dari tiap segmentasi dan *average pooling* yang menghitung rata-rata nilai dari setiap segmentasi (Bowo, dkk., 2020).

Fully connected layer adalah salah satu komponen penting dalam arsitektur *neural network*, termasuk CNN. Dalam CNN, *Fully connected layer* digunakan untuk menggabungkan informasi dari banyak *input* dengan banyak *output*. Lapisan ini digunakan pada akhir arsitektur untuk menggabungkan informasi dari *convolutional layer* dan mengurangi dimensi gambar. *Fully connected layer* digunakan sebagai bagian dari arsitektur CNN yang digunakan untuk klasifikasi gambar (Ding, dkk., 2021).

2.4 Python

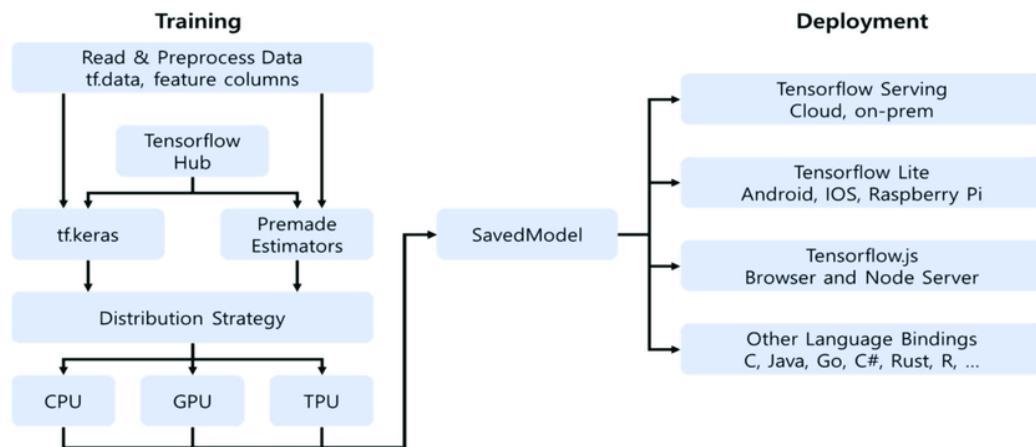
Python adalah bahasa pemrograman yang dapat digunakan di berbagai sistem operasi. Fleksibilitasnya membuatnya menjadi pilihan utama bagi para pengembang perangkat lunak di seluruh dunia. Tidak peduli apakah Anda menggunakan Windows, macOS, Linux, atau sistem operasi lainnya, Python dapat dijalankan dengan lancar dan efisien di mana saja. Python diciptakan oleh Guido van Rossum pada awal tahun 1990-an di Institut Riset Nasional untuk Matematika dan *Computer Science* di Belanda. Guido mengambil nama python dari sebuah serial komedi inggris berjudul "Monty Python Flying Circus". Python dikenalkan ke publik pada tahun 1990 sebagai penerus bahasa pemrograman *All Basic Code (ABC)*.

Python cukup populer di kalangan pengguna baik pemula maupun professional. Hal ini karena Python menggunakan sintaks bahasa Inggris yang sederhana dan konsisten sehingga mudah dipahami dan ditulis, memiliki beragam *library* berisi fungsi-fungsi dan modul-modul yang sudah terdefinisi dan dapat digunakan untuk membangun berbagai jenis aplikasi, mulai dari *web development*, *scientific computing*, *machine learning*, hingga *game development*.

Python juga bekerja dengan baik untuk aplikasi sederhana maupun kompleks. Dengan menggunakan sintaks bahasa Inggris yang sederhana membuat para pengguna dapat fokus kepada pemecahan masalah daripada memikirkan sintaks yang rumit. Kemampuan Python masih dapat diperluas dengan menambahkan *library* pihak ketiga untuk memudahkan pengembang aplikasi (Wibowo, 2011).

2.5 TensorFlow

TensorFlow adalah sebuah *framework open-source* yang dikembangkan oleh *Google Brain Team*. Dengan TensorFlow, para *programmer* dapat dengan mudah membuat berbagai jenis model machine learning, mulai dari jaringan saraf tiruan sederhana hingga arsitektur *deep learning* yang kompleks. *Framework* ini dapat digunakan untuk pengembangan model *machine learning*, yang terdiri dari berbagai komponen yang dapat digunakan secara terpisah atau bersamaan, termasuk layer-layer untuk membangun model, algoritma optimasi untuk melatih model, dan utilitas untuk evaluasi dan pengujian model. Dalam pengembangan model *machine learning*, TensorFlow dapat digunakan dengan bahasa pemrograman Python (Putra, dkk., 2023). Seperti yang ditunjukkan pada Gambar 2.14, arsitektur TensorFlow menyediakan struktur yang komprehensif untuk mendukung pengembangan dan implementasi model-model tersebut.



Gambar 2.14 Arsitektur TensorFlow (Shin & Kim, 2022)

Dengan struktur yang fleksibel TensorFlow memungkinkan para pengembang untuk menguji dan mengimplementasikan berbagai macam algoritma pelatihan yang inovatif. Model yang dibangun menggunakan TensorFlow dapat di-*deploy* ke berbagai platform mulai dari Android hingga *web browser* secara mudah dengan sedikit atau tanpa modifikasi kode. Salah satu contohnya adalah model yang telah dibuat dapat diintegrasikan ke dalam aplikasi android untuk menyediakan fitur kecerdasan buatan yang canggih di telepon genggam, selain itu model juga dapat di-*deploy* ke *server backend* untuk mendukung aplikasi web.

2.6 Android Studio

Android Studio adalah sebuah IDE (*Integrated Development Environment*) yang dikembangkan oleh Google sebagai pengembangan aplikasi Android. Android Studio memiliki fitur-fitur pendukung dalam proses pengembangan aplikasi Android. Salah satu fitur utamanya adalah *system build* berbasis *Gradle* yang fleksibel, memungkinkan pengembang untuk mengatur dependensi proyek dengan lebih efisien. Selain itu, Android Studio dilengkapi dengan emulator yang cepat dan kaya fitur, memungkinkan pengembang untuk mensimulasikan aplikasi mereka secara menyeluruh dalam berbagai skenario.

Android Studio juga memungkinkan pengembang untuk dengan mudah melakukan pengembangan aplikasi pada semua perangkat Android, dari yang sederhana hingga yang kompleks. Fitur "*Apply Changes*" memungkinkan

pengembang melakukan *push* pada perubahan kode dan sumber daya ke aplikasi yang sedang berjalan tanpa perlu memulai ulang aplikasi. Selain itu, Android Studio juga menyediakan *template* kode dan integrasi GitHub yang membantu pengembang dalam membuat fitur-fitur aplikasi umum dan mengimpor kode sampel dengan mudah.

Android Studio memiliki *framework* dan alat pengujian yang lengkap, hal tersebut memungkinkan pengembang untuk menguji aplikasi mereka dengan baik sebelum diluncurkan. Selain itu, Android Studio juga mendukung bahasa pemrograman C++ dan NDK, memberikan fleksibilitas yang lebih besar dalam pengembangan aplikasi yang memerlukan kinerja tinggi dan integrasi dengan kode-kode yang sudah ada sebelumnya (Anggraini, Danuri, & Tedyyana, 2018).

BAB III

METODE PENELITIAN

3.1 Tempat dan Waktu Penelitian

Penelitian dilakukan di Laboratorium Elektronika dan Instrumentasi, Departemen Fisika, FSM, UNDIP, Semarang. Penelitian dilaksanakan dari bulan Maret 2024 sampai dengan bulan Juni 2024.

3.2 Alat dan Bahan Penelitian

Pada penelitian ini digunakan alat diantaranya laptop, dan bahasa pemrograman Python, Kotlin, dan java. Adapun rincian spesifikasi Laptop yang digunakan pada penelitian ini yaitu AMD Ryzen 5 5600U, RAM 16GB, ROM SSD 512GB, GPU Radeon Integrated, dan OS Windows 11. Pemrograman akan dilakukan menggunakan Bahasa pemrograman python dengan open-source library seperti TensorFlow, NumPy, Matplotlib, dan KERAS. Bahasa pemrograman Java dan Kotlin untuk mengembangkan aplikasi berbasis android menggunakan IDE Android Studio.

Bahan yang digunakan pada penelitian ini meliputi dataset citra daun tomat yang terjangkit penyakit dan yang sehat. Dataset yang digunakan pada penelitian ini diperoleh dari website <https://www.kaggle.com/datasets/jarvis705/tomato-leaf-disease>, data tersebut terdiri dari 10 kelas yaitu kelas daun tanaman tomat yang sehat dan 9 kelas penyakit tanaman tomat dengan rincian *Tomato mosaic virus*, *Target Spot*, *Bacterial spot*, *Tomato Yellow Leaf Curl Virus*, *Late blight*, *Leaf Mold*, *Early blight*, *Spider mites*, *Tomato healthy*, *Septoria leaf spot*. Tabel keterangan dataset yang digunakan dalam penelitian ini dapat dilihat dalam Tabel 3.1.

Tabel 3.1 Dataset citra daun tomat

Kelas	Train	Validation	Test
<i>Tomato mosaic virus</i>	408	56	56
<i>Target Spot</i>	403	45	45
<i>Bacterial Spot</i>	374	52	52
<i>Tomato Yellow Leaf Curl Virus</i>	406	48	48
<i>Late Blight</i>	400	53	53
<i>Leaf Mold</i>	414	59	59
<i>Early Blight</i>	401	43	43
<i>Spider Mites</i>	379	53	53
<i>Septoria Leaf Spot</i>	399	55	55
<i>Healthy</i>	394	56	56
Total	3978	520	520

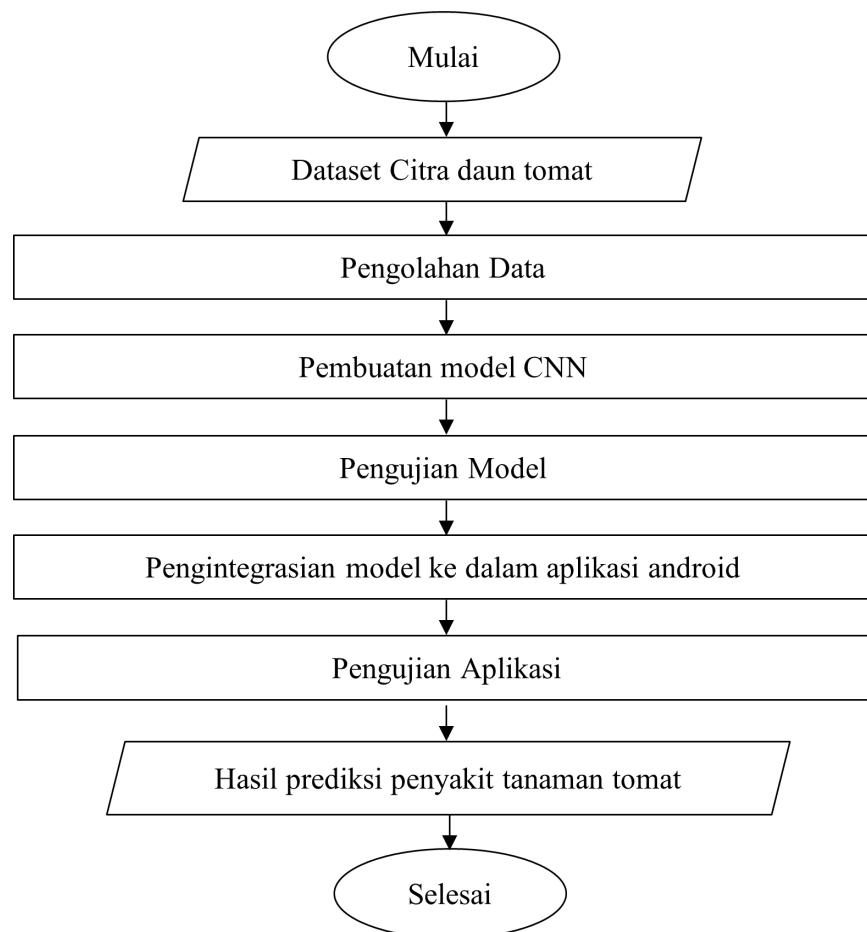
3.3 Prosedur Penelitian

Langkah pertama dalam penelitian ini adalah membagi dataset citra tanaman tomat menjadi tiga bagian, yaitu data latih (*training*), data validasi (*validation*), dan data uji (*test*). Data ini kemudian dimasukkan ke dalam folder database. Selanjutnya, dilakukan proses pra-pemrosesan gambar pada 10 kelas citra tanaman tomat.

Kemudian dilakukan pembuatan model yang dapat mengidentifikasi jenis penyakit tanaman tomat. Pembuatan arsitektur model dilakukan dengan pengidentifikasian, perancangan, pelatihan dan validasi. Model yang dirancang menggunakan arsitektur CNN dengan menentukan jumlah lapisan *convolutional*, *pooling*, dan *fully-connected* yang sesuai untuk mendapatkan hasil akhir yang diinginkan. Kemudian dilanjutkan dengan menentukan banyaknya filter, jumlah lapisan konvolusi, fungsi aktivasi, *optimizer*, *callback* yaitu *ReduceLROnPlateau* serta *EarlyStopping* dan *dropout rate* untuk memastikan hasil pelatihan dan validasi dari model tidak mengalami *overfitting* ataupun *underfitting*. Hasil akhir dari proses

ini berupa model yang dapat mengidentifikasi penyakit pada tanaman tomat. Setelah membuat model yang dapat mengidentifikasi penyakit pada tanaman tomat perlu dilakukan pengujian terhadap model tersebut untuk mengetahui performa model pada citra yang belum pernah dilihat sebelumnya. Pengujian ini dilakukan dengan mengevaluasi model menggunakan data *testing* dengan matriks evaluasi. Pada penelitian ini digunakan *confusion matrix* sebagai matriks evaluasi. Dengan *confusion matrix* kita dapat mengetahui bagaimana model dapat mengidentifikasi data dengan benar ke dalam kategori yang sesuai.

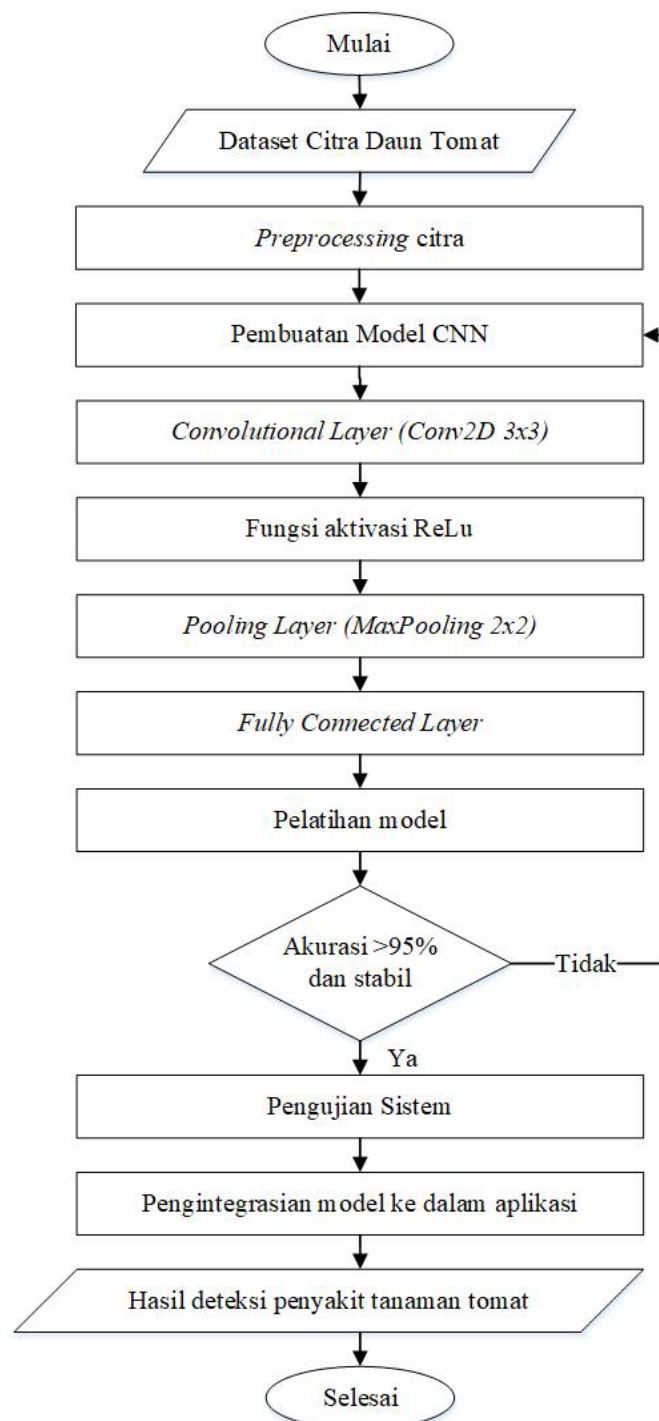
Setelah dilakukan pengujian pada model, maka akan dilakukan pengintegrasian model ke dalam aplikasi android menggunakan *Android Studio* sehingga aplikasi dapat digunakan untuk mengidentifikasi penyakit tanaman tomat secara *real-time*. Diagram blok penelitian dapat dilihat dalam gambar 3.1.



Gambar 3.1 Diagram Blok Penelitian

3.4 Diagram Alir

Prosedur penelitian ini menggunakan CNN yang diilustrasikan dengan diagram alir pada Gambar 3.2



Gambar 3.1 Diagram Alir

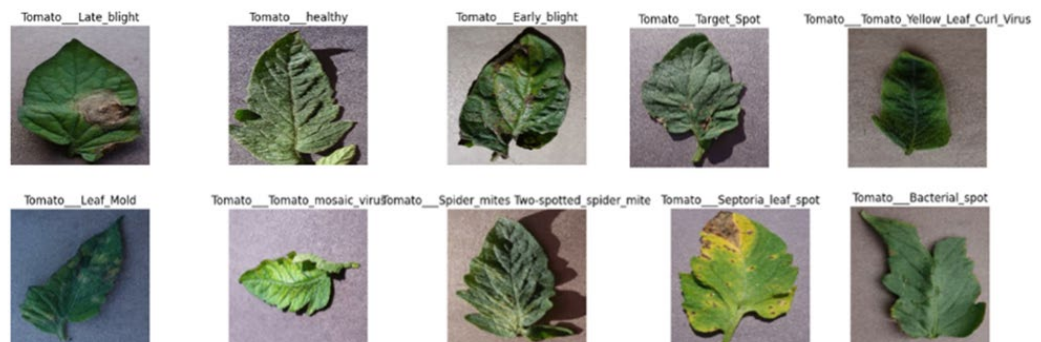
BAB IV

HASIL DAN PEMBAHASAN

Pada bab ini akan disajikan hasil dan pembahasan penelitian yang meliputi pengembangan model *Convolutional Neural Network* (CNN) untuk mendeteksi penyakit pada daun tomat serta implementasi model tersebut dalam sebuah aplikasi *android*.

4.1 Pengolahan Data

Proses pembuatan dan pengujian model dengan menggunakan *convolutional neural network* ini dimulai dengan mengumpulkan dataset citra daun tomat. Sampel citra tanaman tomat yang digunakan pada penelitian ini dapat dilihat pada Gambar 4.1



Gambar 4.1 Dataset Citra Tanaman Tomat

Citra yang telah dikumpulkan kemudian melalui tahap *preprocessing* untuk memastikan citra dapat diolah dengan baik. Pada tahap ini, citra dibagi menjadi data latih, data validasi, dan data uji. Selain itu, citra juga diubah ukurannya (*resize*) menjadi 256x256 piksel dan disesuaikan skala nilainya (*rescale*) untuk memastikan semua input memiliki ukuran yang seragam.

4.2 Hasil Pelatihan Model

Untuk mencapai tingkat akurasi yang optimal, berbagai uji coba model telah dilakukan. Uji coba ini melibatkan variasi beberapa parameter serta penggunaan jumlah lapisan konvolusi yang berbeda.

4.1.1 Pelatihan Model dengan 3 Lapisan Konvolusi Tanpa *Reduce Learning Rate* dan *Early Stopping*

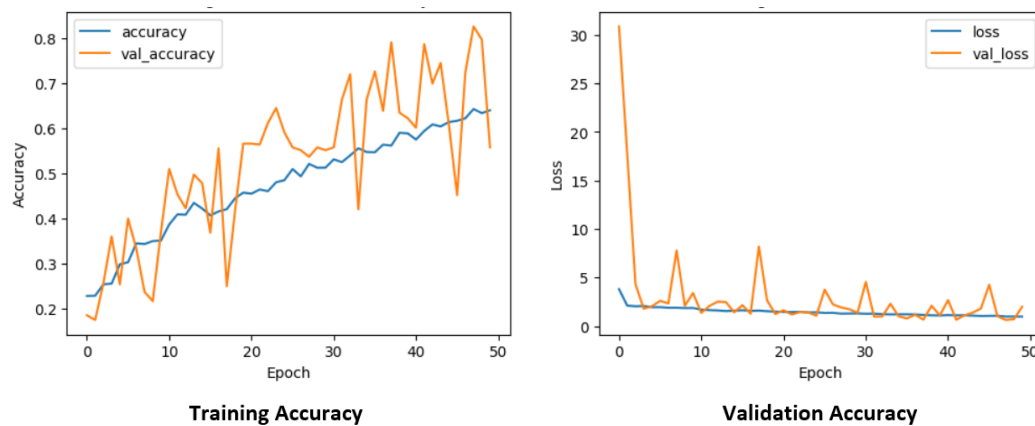
Model yang pertama dirancang terdiri dari 3 lapisan konvolusi tanpa menggunakan *callback reduce learning rate* dan *early stopping*. Struktur model ini terdiri dari *input layer* dengan ukuran citra 256x256 piksel dan tiga saluran warna RGB. Selanjutnya terdapat 3 lapisan konvolusi dengan kernel berukuran 3x3 untuk mengekstrak fitur citra dengan jumlah filter berturut-turut 16, 32, dan 64 pada setiap lapisan untuk menangkap tingkat fitur yang berbeda. Setiap *Convolutional Layer* diakhiri dengan *batch normalization* untuk memastikan setiap nilai input ke lapisan selanjutnya memiliki distribusi yang lebih stabil. Kemudian dilanjutkan dengan *pooling layer* di mana pada penelitian ini digunakan *Max Pooling 2D* dengan ukuran 2x2 untuk mengecilkan dimensi fitur dan mengurangi *overfitting*. Lapisan selanjutnya adalah *flatten layer*, lapisan ini mengubah citra multi-dimensi menjadi satu dimensi (vektor) untuk meratakan fitur-fitur yang telah diekstrak sebelumnya sehingga mempermudah pemrosesan citra.

Lapisan *flatten* ini kemudian diikuti oleh lapisan *dense* atau diketahui juga sebagai *fully connected layer* dengan 128 *neuron* dan fungsi aktivasi ReLU. Selanjutnya, untuk lebih meningkatkan generalisasi model dan mencegah *overfitting*, diterapkan lapisan *dropout* dengan tingkat drop sebesar 0.5. Terakhir, output layer menggunakan *dense layer* dengan jumlah *neuron* sesuai dengan kelas berjumlah 10, dan fungsi aktivasi *softmax*. Fungsi aktivasi *softmax* digunakan pada *output layer* untuk mengubah *output* dari model menjadi distribusi probabilitas yang mewakili kemungkinan prediksi untuk setiap kelas. Detail model dan jumlah parameter dapat dilihat pada Tabel 4.1.

Tabel 4.1 Model pelatihan pertama

Layer (type)	Output Shape	Param
Sequential (Sequential)	(None, 256,256,3)	0
Conv2d (Conv2D)	(None, 254,254,16)	448
Batch_normalization (batchNormalization)	(None, 254,254,16)	64
Max_pooling2d (MaxPooling2D)	(None, 127,127,16)	0
Conv2d_1 (Conv2D)	(None, 125,125,32)	4640
Batch_normalization_1 (batchNormalization)	(None, 125,125,32)	128
Max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 32)	0
Conv2d_2 (Conv2D)	(None, 60, 60, 64)	18496
Batch_normalization_2 (batchNormalization)	(None, 60, 60, 64)	256
Max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 64)	0
Flatten (flatten)	(None, 57600)	0
Dense (dense)	(None, 128)	7372928
Dropout (dropout)	(None, 128)	0
Dense_1 (dense)	(None, 10)	1290

Hasil pelatihan citra menggunakan model pada Tabel 4.1 menunjukkan bahwa model dapat mencapai akurasi 0.6408 dengan *loss* 0.9774 pada data latih. Untuk data validasi, model mencapai akurasi 0.5583 dengan *loss* 1.9872. Grafik yang menunjukkan akurasi dan *loss* pada data latih serta validasi selama proses pelatihan dapat dilihat pada Gambar 4.2.



Gambar 4.2 Grafik hasil pelatihan dengan 3 lapisan konvolusi

Berdasarkan grafik hasil penelitian pada Gambar 4.2 terlihat bahwa model belum mencapai tingkat akurasi terbaiknya. Hal ini disebabkan oleh jumlah lapisan konvolusi yang terbatas, hanya terdiri dari 3 lapisan, sehingga model tidak mampu untuk menangkap fitur-fitur kompleks pada citra. Selain itu, perbedaan yang signifikan antara grafik pelatihan dan validasi menunjukkan adanya *overfitting* pada model. Grafik validasi menunjukkan kinerja yang lebih rendah dibandingkan grafik pelatihan, mengindikasikan bahwa model tidak dapat menggeneralisasi dengan baik ke data yang tidak terlihat sebelumnya.

4.1.2 Pelatihan Model dengan Penambahan Lapisan Konvolusi dan Regularisasi

Setelah mengevaluasi hasil pengujian sebelumnya, terlihat bahwa akurasi latihan model masih belum mencapai tingkat akurasi terbaiknya, dan model mengalami *overfitting*. Kemudian beberapa parameter model ditambahkan untuk mendapatkan hasil yang lebih optimal. Hal pertama yang ditambahkan adalah lapisan konvolusi dengan jumlah filter 128 dengan tujuan meningkatkan kemampuan model agar dapat menangkap fitur-fitur kompleks pada citra. Untuk mengatasi masalah *overfitting*, diterapkan regularisasi L2 dengan faktor 0.01. Regularisasi ini digunakan untuk mengendalikan kompleksitas model dan meminimalisir resiko model terlalu fokus pada detail pelatihan yang spesifik supaya

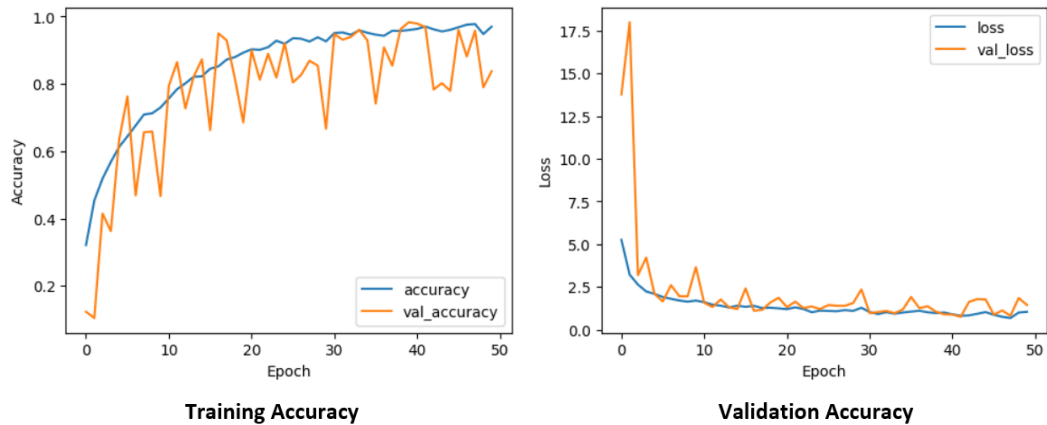
lebih stabil dan akurat dalam mengidentifikasi data baru dan evaluasi terhadap data validasi. Detail rancangan model CNN yang telah diperbarui dapat dilihat dalam Tabel 4.2.

Tabel 4.2 Model CNN yang telah diperbarui

Layer (type)	Output Shape	Param
Sequential (Sequential)	(None, 256,256,3)	0
Conv2d (Conv2D)	(None, 254,254,16)	448
Batch_normalization (batchNormalization)	(None, 254,254,16)	64
Max_pooling2d (MaxPooling2D)	(None, 127,127,16)	0
Conv2d_1 (Conv2D)	(None, 125,125,32)	4640
Batch_normalization_1 (batchNormalization)	(None, 125,125,32)	128
Max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 32)	0
Conv2d_2 (Conv2D)	(None, 60, 60, 64)	18496
Batch_normalization_2 (batchNormalization)	(None, 60, 60, 64)	256
Max_pooling2d_3 (MaxPooling2D)	(None, 30, 30, 64)	0
Conv2d_3 (Conv2D)	(None, 28, 28, 128)	73856
Batch_normalization_3 (batchNormalization)	(None, 28, 28, 128)	512
Flatten (flatten)	(None, 25088)	0
Dense (dense)	(None, 128)	3211392
Dropout (dropout)	(None, 128)	0
Dense_1 (dense)	(None, 10)	1290

Dengan menggunakan model yang telah diperbarui, hasil pelatihan citra menunjukkan peningkatan signifikan dalam akurasi. Model ini mencapai akurasi sebesar 0.9701 pada data pelatihan dan 0.8375 pada data validasi, dengan nilai *loss* sebesar 1.0350 untuk data pelatihan dan 1.4348 untuk data validasi. Grafik yang

menunjukkan akurasi dan *loss* pada data latih serta validasi selama proses pelatihan 50 *epoch* dapat dilihat pada Gambar 4.3

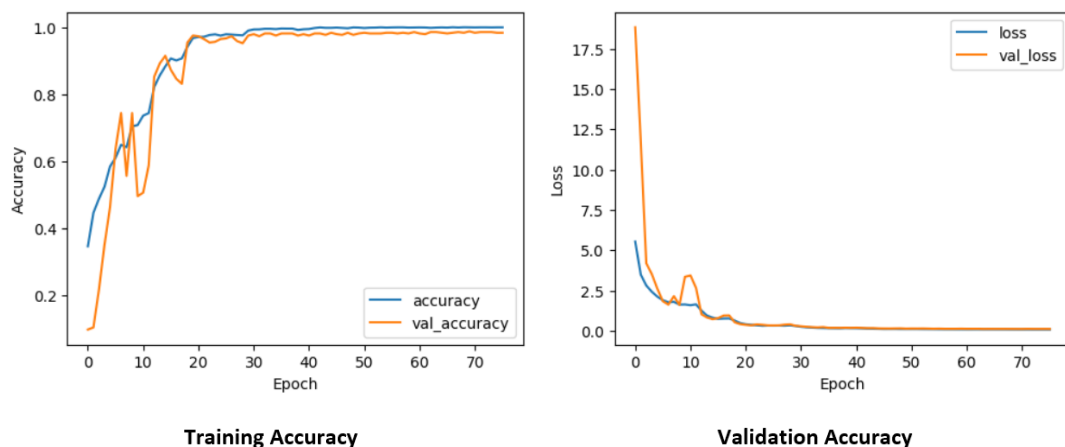


Gambar 4.3 Grafik hasil pelatihan dengan penambahan lapisan konvolusi dan regularisasi

Berdasarkan grafik hasil pelatihan pada Gambar 4.3, terlihat bahwa ada kenaikan yang stabil dalam akurasi data latih karena adanya penambahan lapisan konvolusi. Namun, grafik akurasi validasi menunjukkan fluktuasi yang cukup signifikan. Meskipun terdapat peningkatan pada beberapa titik, fluktuasi ini menandakan bahwa model belum optimal dalam melakukan generalisasi yang baik terhadap data yang belum pernah dilihat sebelumnya. Ini mengindikasikan bahwa meskipun penambahan lapisan konvolusi membantu model untuk menangkap fitur-fitur kompleks pada citra, masih diperlukan upaya lebih lanjut untuk meningkatkan kemampuan model dalam melakukan generalisasi terhadap data validasi. Penambahan regularisasi L2 pada data latih juga membantu dalam mempertahankan stabilitas model dengan membatasi kompleksitasnya, serta mendorong generalisasi yang lebih baik pada data yang belum pernah dilihat sebelumnya. Jika dibandingkan dengan Gambar 4.2 fluktuasi yang terjadi lebih kecil. Namun, penambahan generalisasi L2 pada model belum cukup menstabilkan laju pembelajaran model sehingga, model masih perlu diperbaiki.

4.1.3 Pelatihan Model dengan Penambahan *Reduce Learning Rate*

Setelah mengamati pengujian sebelumnya, model menunjukkan performa akurasi yang meningkat. Akan tetapi, masih terdapat fluktuasi pada grafik validasi yang cukup signifikan. Hal ini menandakan model belum mampu melakukan generalisasi dengan baik terhadap data baru. Pada uji coba ini diterapkan struktur model yang sama, akan tetapi untuk mengoptimalkan performa model ditambahkan *callback ReduceLROnPlateau*. *Callback* ini berguna untuk mengurangi laju pembelajaran sehingga model dapat belajar dengan lebih stabil dan menghasilkan akurasi yang lebih baik. Grafik yang menunjukkan akurasi dan *loss* pada data latih serta validasi selama proses pelatihan 76 *epoch* dapat dilihat dalam Gambar 4.4.



Gambar 4.4 Grafik hasil pelatihan dengan penambahan *reduce learning rate*

Berdasarkan Gambar 4.4 terlihat bahwa dalam pelatihan model ini, pada awal *epoch* model masih kesulitan untuk mengidentifikasi citra yang belum pernah dilihat sebelumnya. Hal ini dapat terlihat karena masih terdapat fluktuasi pada grafik validasi akurasi, serta nilai akurasi yang masih rendah. Namun, seiring berjalannya *epoch* terlihat adanya peningkatan yang konsisten dalam performa model. Pada *epoch* ke-12 *ReduceLROnPlateau* pertama kali diterapkan oleh model untuk mengurangi laju pembelajaran. Penggunaan *ReduceLROnPlateau* terbukti efektif dalam menstabilkan proses pembelajaran. Ketika laju pembelajaran dikurangi, model menjadi lebih stabil dan menunjukkan peningkatan yang lebih

baik pada akurasi validasi. Peningkatan yang stabil terus terlihat hingga akhir proses pelatihan. Pada akhir pelatihan diperoleh akurasi sebesar 0.9995 pada data pelatihan dan 0.9833 pada data validasi, dengan nilai *loss* sebesar 0.0759 untuk data pelatihan dan 0.1099 untuk data validasi.

4.3 Pembahasan

Dari serangkaian pelatihan model yang telah dilakukan, dapat terlihat bahwa penambahan lapisan konvolusi dan penerapan regularisasi serta *callback ReduceLROnPlateau* memberikan dampak signifikan terhadap peningkatan performa model dalam mengklasifikasikan citra. Pada model pertama yang hanya menggunakan tiga lapisan konvolusi, akurasi pada data latih hanya mencapai 0.6408 dengan nilai *loss* 0.9774, sementara pada data validasi akurasi hanya mencapai 0.5583 dengan nilai *loss* 1.9872. Hal ini menunjukkan bahwa model mengalami *overfitting* dan belum mampu menangkap fitur-fitur kompleks dari citra secara optimal. Peningkatan performa model terlihat jelas pada pelatihan kedua setelah penambahan satu lapisan konvolusi dengan jumlah filter 128 dan penerapan regularisasi L2. Hasil pelatihan menunjukkan akurasi sebesar 0.9701 pada data latih dan 0.8375 pada data validasi, dengan nilai *loss* masing-masing sebesar 1.0350 dan 1.4348. Meskipun begitu, masih terdapat fluktuasi yang signifikan pada grafik akurasi validasi, menunjukkan bahwa model masih perlu perbaikan lebih lanjut dalam hal generalisasi.

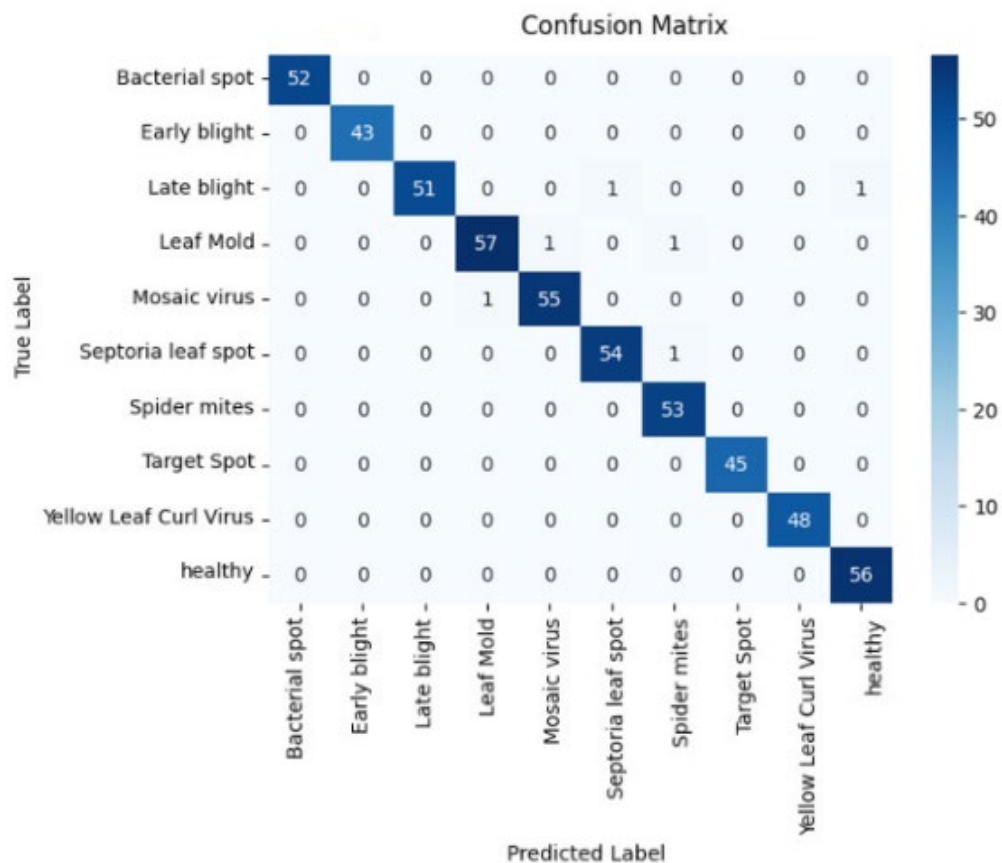
Penggunaan *callback ReduceLROnPlateau* pada pelatihan berikutnya berhasil mengurangi fluktuasi pada grafik akurasi validasi. Karena dengan penggunaan *callback ReduceLROnPlateau* jika setiap 3 epoch pelatihan nilai *loss* tidak menunjukkan penurunan maka laju pembelajaran akan otomatis dikurangi sebesar 50% yang membantu model belajar lebih baik. Selain itu, pada pelatihan terakhir model juga diberi epoch yang lebih banyak sehingga mendapatkan waktu lebih agar dapat memahami data dengan maksimal. Pada akhir pelatihan, model mencapai akurasi 0.9995 pada data pelatihan dan 0.9833 pada data validasi, dengan nilai *loss* sebesar 0.0759 untuk data pelatihan dan 0.1099 untuk data validasi.

4.4 Pengujian

Sistem diuji untuk mengukur bagaimana performanya ketika digunakan dalam dunia nyata. Pengujian dilakukan menggunakan data uji menggunakan *confusion matrix*. Nilai akurasi diperoleh dari persamaan 4.1

$$\frac{T_c - E}{T_c} \times 100\% \quad 4.1$$

Di mana T_c adalah total sampel citra teridentifikasi benar, E error atau jumlah kesalahan dalam identifikasi. Hasil *confusion matrix* data *test* dapat dilihat pada Gambar 4.6




Gambar 4.5 *Confusion Matrix* Data Uji

Berdasarkan *confusion matrix* pada Gambar 4.5, untuk 520 data uji citra daun tomat, sistem identifikasi menunjukkan kinerja yang sangat baik dalam mengidentifikasi berbagai penyakit dan kondisi daun tomat dengan Tingkat rata-

rata akurasi yang tinggi. Dari total 520 sampel yang diuji, 514 sampel berhasil diidentifikasi dengan benar, menghasilkan tingkat akurasi keseluruhan sebesar 98.8%. Citra yang gagal diidentifikasi dapat dilihat pada Tabel 4.3.

Tabel 4.3 Citra yang gagal diidentifikasi

No.	Citra	Citra Seharusnya	Citra Teridentifikasi	Penyebab Kegagalan
1		<i>Late Blight</i>	<i>Septoria Leaf Spot</i>	Terdapat kemiripan gejala pada kedua penyakit yaitu bercak kecoklatan
2		<i>Late Blight</i>	<i>Healthy</i>	Terdapat gejala <i>late blight</i> yang kurang jelas
3		<i>Leaf Mold</i>	<i>Mosaic Virus</i>	Adanya perubahan warna hijau tua seperti gejala <i>mosaic virus</i>
4		<i>Leaf Mold</i>	<i>Spider Mites</i>	Terdapat gejala bitnik bitnik yang menyerupai spider mites
5		<i>Mosaic Virus</i>	<i>Leaf Mold</i>	Adanya perubahan warna pada pinggir daun menyerupai <i>leaf mold</i>
6		<i>Septoria Leaf Spot</i>	<i>Spider Mites</i>	Adanya bintik pada daun yang menyerupai spider mites

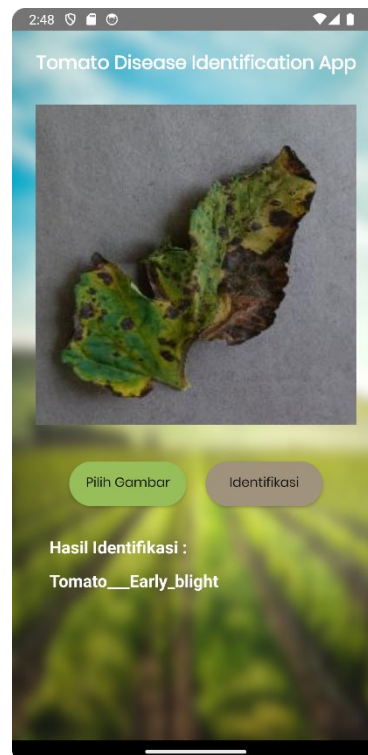
Pada uji citra tanaman daun tomat yang terjangkit *Bacterial Spot*, *Early Blight*, *Target Spot*, *Yellow Leaf Curl Virus*, dan *Healthy*, masing-masing dari 52, 43, 45, 48, dan 56 citra uji semuanya berhasil diidentifikasi dengan benar. Untuk

tanaman yang terjangkit *Late Blight*, dari 53 kasus uji, 51 di antaranya berhasil diidentifikasi dengan benar, sementara 2 kasus menghasilkan kesalahan identifikasi sebagai *Septoria Leaf Spot* dan *Healthy*. Pada pengujian citra tanaman yang terjangkit *Leaf Mold*, dari 59 kasus uji terdapat 2 kesalahan identifikasi, yaitu teridentifikasi sebagai *Mosaic Virus* dan *Spider Mites*. Untuk pengujian *Mosaic Virus* pada 56 citra, terdapat 1 kesalahan identifikasi, yaitu teridentifikasi sebagai *Leaf Mold*. Pada pengujian tanaman yang terjangkit *Septoria Leaf Spot*, juga terdapat 1 kesalahan identifikasi, yaitu teridentifikasi sebagai *Spider Mites*. Hasil evaluasi keseluruhan menunjukkan performa model pada data latih sangat baik.

Berdasarkan penelitian sebelumnya yang dilakukan oleh Shamima Parvez dkk. pada tahun 2023 digunakan citra daun tomat dengan 2 jenis penyakit yaitu *bacterial spot*, *yellow leaf curl virus* dan daun tomat sehat. Dari penelitian tersebut diperoleh nilai akurasi mencapai 0.9839 untuk data pelatihan dengan *loss* sebesar 0.0495 dan *validation accuracy* 0.9807 serta 0.0434 untuk *validation loss* yang diperoleh dari 30 epoch. Maka pada penelitian ini saya memperluas cakupan dengan menggunakan 5000 citra yang terdiri dari 9 jenis penyakit yaitu *Tomato mosaic virus*, *Target Spot*, *Bacterial spot*, *Tomato Yellow Leaf Curl Virus*, *Late blight*, *Leaf Mold*, *Early blight*, *Spider mites*, dan *Septoria leaf spot* dan juga citra daun tomat sehat. Pada penelitian ini diperoleh akurasi yang lebih tinggi yaitu 0.9995 untuk data pelatihan dengan *loss* 0.0759 dan 0.9833 untuk *validation accuracy* dengan *loss* 0.1099. Pada penelitian ini dapat diperoleh nilai akurasi yang lebih tinggi karena model melakukan pelatihan selama 76 epoch yang lebih lama dibandingkan dengan penelitian sebelumnya yang hanya 30 epoch. Jumlah epoch yang lebih banyak memberikan model waktu lebih lama untuk mempelajari data dan mengurangi *overfitting* pada data pelatihan. Penelitian ini juga mengimplementasikan model dalam aplikasi android sehingga dapat membantu identifikasi penyakit tanaman tomat secara *real-time* yang membedakannya dengan penelitian terdahulu.

4.5 Implementasi Model Dalam Aplikasi Android

Setelah model diuji dan sudah mencapai nilai akurasi yang diharapkan, proses selanjutnya adalah menyimpan model tersebut ke dalam bentuk *TensorFlow Lite* (*TFLite*) sehingga dapat diintegrasikan ke dalam aplikasi android serta mengoptimalkan model untuk dijalankan pada perangkat android. Proses konversi model ke *TFLite* dilakukan menggunakan *TensorFlow Lite Converter*, yang mengubah model dari format .h5 ke format *TFLite*. Pembuatan aplikasi *android* dilakukan menggunakan IDE *android studio*. Proses ini dimulai dengan merancang *User Interface* yang terdiri dari fitur memilih foto untuk dideteksi, tombol deteksi dan hasil deteksi. Setelah UI dirancang, langkah berikutnya adalah mengimpor model *TFLite* ke dalam proyek *Android Studio*. Sehingga fitur deteksi pada aplikasi dapat dijalankan. Tampilan aplikasi pada Sistem Operasi android pada penelitian ini dapat dilihat pada Gambar 4.6.



Gambar 4.6 Tampilan UI aplikasi pada Sistem Operasi android

Fitur aplikasi ini terdiri dari tombol untuk mengakses galeri pengguna sehingga pengguna dapat memilih foto yang ingin dideteksi, tombol identifikasi untuk memicu model yang telah diimpor ke dalam aplikasi dan teks untuk menampilkan hasil identifikasi model. Penggunaan aplikasi dirancang sederhana untuk memudahkan pengguna dalam mendeteksi penyakit pada tanaman tomat secara cepat dan akurat. Penggunaan aplikasi dimulai dengan mengklik tombol “pilih gambar”, tombol tersebut akan mengakses galeri pengguna kemudian pengguna dapat memilih citra tanaman tomat yang ingin di deteksi jenis penyakitnya. Setelah itu pengguna dapat mengklik tombol identifikasi dan model akan secara otomatis mendeteksi penyakit yang ada pada citra yang telah dipilih. Hasil identifikasi model kemudian akan muncul dibawah kedua tombol tersebut.

BAB V

KESIMPULAN

5.1 Kesimpulan

Dari penelitian ini dapat disimpulkan bahwa aplikasi yang dikembangkan memperoleh akurasi terbaik model dengan nilai akurasi 0.9995 untuk data pelatihan dengan *loss* 0.0759 dan 0.9833 untuk *validation accuracy* dengan *loss* 0.1099 serta akurasi pengujian diperoleh 98.8% untuk 520 citra uji dari 9 jenis penyakit yaitu *Tomato mosaic virus*, *Target Spot*, *Bacterial spot*, *Tomato Yellow Leaf Curl Virus*, *Late blight*, *Leaf Mold*, *Early blight*, *Spider mites*, *Tomato healthy*, *Septoria leaf spot* dan juga untuk tanaman tomat yang sehat. Hasil ini menunjukkan bahwa aplikasi ini dapat digunakan untuk membantu para petani untuk mengurangi resiko gagal panen melalui deteksi dini penyakit pada tanaman tomat.

5.2 Saran

Berdasarkan hasil penelitian, penulis menyarankan untuk penelitian selanjutnya mempertimbangkan penggunaan dataset yang lebih banyak untuk meningkatkan cakupan deteksi penyakit yang lebih luas. Selain itu, disarankan untuk mengembangkan aplikasi dengan menambahkan fitur-fitur tambahan yang dapat meningkatkan fungsionalitas dan kegunaan aplikasi tersebut.

DAFTAR PUSTAKA

- Acosta, N., & Cañas, L. A. (2019). Evaluation of Trabon for the Control of Two Spotted Spider Mites, *Tetranychus urticae*, Infesting Indeterminate Greenhouse Tomatoes, 2017. *Arthropod Management Tests*, 44, 1-3.
- Alwanda, M. R., Ramadhan, R. K., & Alamsyah, D. (2020). Implementasi Metode Convolutional Neural Network Menggunakan Arsitektur LeNet-5 untuk Pengenalan Doodle. *Jurnal Algoritme*, 1(1), 45-56.
- Azmi, K., Defit, S., & Sumijan. (2023). Implementasi Convolutional Neural Network (CNN) Untuk Klasifikasi Batik Tanah Liat Sumatera Barat. *Jurnal Unitek*, 16(1), 28-40.
- Baideng, E. L. (2016). Kelompok Tani Tomat dalam Penerapan Pengendalian Hama Terpadu di Desa Kakaskasen III untuk Memantapkan Produksi dan Meningkatkan Pendapatan Petani. *Jurnal LPPM Bidang Sains dan Teknologi*, 3(1), 34-43.
- Bowo, T. A., Syaputra, H., & Akbar, M. (2020). Penerapan Algoritma Convolutional Neural Network Untuk Klasifikasi Motif Citra Batik Solo. *Journal of Software Engineering Ampera*, 1(2), 82-96.
- Chaudhari, R., Marathe, R., Rane, R., & Pingle, V. (2019). Detection of Disease in Tomato Plants Using Deep Learning. *ICSD 2019*, 9(4), 525-540.
- Choudhary, A., Rishi, R., Dhaka, V., & Ahlawat, S. (2010). Influence of Introducing an Additional Hidden Layer on the Character Recognition Capability of a BP Neural Network having One Hidden Layer. *International Journal of Engineering and Technology*, 2(1), 24-28.
- Courtney, J. (2001). Application of Digital Image Processing to Marker-free Analysis of Human Gait. *Measurement Science Review*, 1, 11-14.
- Devi, P. A., & Rosyid, H. (2022). Pemaparan Materi Dasar Pengolahan Citra Digital untuk Upgrade Wawasan. *Jurnal Abdi Masyarakat Indonesia (JAMSI)*, 2, 1259-1264.
- Ding, X., Xia, C., Zhang, X., Han, J., & Ding, G. (2021). RepMLP: Re-parameterizing Convolutions into Fully-connected Layers for Image Recognition. *ArXiv*.
- Ersyad, M. Z., Ramadhani, K. N., & Arifianto, A. (2020). Pengenalan Bentuk Tangan dengan Convolutional Neural Network (CNN). *e-Proceeding of Engineering*, 7, 8212-8222.
- Hadi, A. S. (2023). Khasiat Buah Tomat (*Solanum lycopersicum*) Berpotensi Sebagai Obat Berbagai Jenis Penyakit. *Journal of Progressive Science and Mathematics*, 1, 7-15.
- Indra, J., Agani, N., & Handayani, H. (2018). Klasifikasi Fertilitas Telur Itik Dengan Pengolahan Citra Digital Menggunakan Raspberry Pi. *Jurnal Ilmu Komputer dan Teknologi Informasi*, 3, 68-76.
- Kamei, A., Dutta, S., Sarker, K., Das, S., Datta, G., & Golda, S. (2018). Target leaf spot of tomato incited by *Corynesporacassiicola*, an emerging disease in

- tomatoproduction under Gangetic alluvial region of WestBengal, India. *Archives of Phytopathology and Plant Protection*, 51, 1039-1048.
- Kumar, P., Luo, S., & Shaukat, K. (2023). A Comprehensive Review of Deep Learning Approaches for Animal Detection on Video Data. (*IJACSA*) *International Journal of Advanced Computer Science and Applications*, 14, 1420-1437.
- Kumaresan, S., Aultrin, K., Kumar, S., & Anand, M. (2021). Transfer Learning With CNN for Classification of Weld Defect. *Digital Object Identifier* 10.1109.
- Latorre, B., & Besoain, X. (2007). Occurrence of Severe Outbreaks of Leaf Mold Caused by *Fulvia fulva* in Greenhouse Tomatoes in Chile. *APS Publications*, 84, 694.
- Li, J., Wang, J.-c., Dong, T.-b., & Chu, D. (2021). Synergistic Effects of a Tomato chlorosis virus and Tomato yellow leaf curl virus Mixed Infection on Host Tomato Plants and the Whitefly Vector. *Frontiers*, 12.
- Mabvakure, B., Martin, D. P., Kraberger, S., Cloete, L., Brunschot, V. S., Geering, A. D., . . . Harkins, G. (2016). Ongoing geographical spread of Tomato yellow leaf curl virus. *Virology*, 498, 257-264.
- Majid, A. M., Khairat, U., & Qaslim, A. (2022). Identifikasi Kualitas Fisik Pada Biji Kopi Menggunakan Teknologi Pengolahan Citra Dengan Metode Neural Network. *Journal Pegguruang: Conference Series*, 4, 12-16.
- Marks, M. (2017, February 02). *University of Wisconsin-Madison*. Retrieved from Wisconsin Horticulture.
- Mohamed, S. I. (2020). Potato Leaf Disease Diagnosis and Detection System Based on Convolution Neural Network. *International Journal of Recent Technology and Engineering (IJRTE)*, 9(4), 254-259.
- Mrkvová, M., Hančinský, R., Grešíková, S., Kaňuková, Š., Ján, B., Miroslav, G., . . . Mihálik, D. (2022). Evaluation of New Polyclonal Antibody Developed for Serological Diagnostics of Tomato Mosaic Virus. *Viruses*, 14, 1331.
- Murya, Y. (2014). *Pemrograman Android Black Box*. Surabaya: Jasakom.
- Muzahardin, Y. S., Fauzi, A., & Nurhayati. (2022). Perbaikan Citra Digital Pada Foto dengan Menggunakan Metode Retinex. *Jurnal Teknik Informatika Kaputama (JTik)*, 6(1), 133-139.
- Neves, A. J., Barbosa, B., Dimas, I., & Soares, S. (2018). Analysis of Emotions From Body Postures Based on Digital Imaging.
- Omori, Y., & Shima, Y. (2020). Image Augmentation for Eye Contact Detection Based on Combination of Pre-trained Alex-Net CNN and SVM. *Journal of Computers*, 15(3), 85-97.
- Parvez, S., Uddin, M., Islam, M., Bharman, P., & Talukder, M. (2023). Tomato Leaf Disease Detection Using Convolutional Neural Network. *Research Square*.
- Purmadani, M. (2024, january 18). *radarsurabayabisnis.id*. Retrieved from radarsurabayabisnis.jawapos.com:https://radarsurabayabisnis.jawapos.com/industri-perdagangan/2183775488/harga-tomat-merangkak-naik-ini-penyebabnya

- Putra, I. A., Kartini, K. S., Suyitno, Y. K., Sugiarta, I., & Puspita, N. M. (2023). Penerapan Library TensorFlow, Cvzone, dan Numpy pada Sistem Deteksi Bahasa Isyarat Secara Real Time. *Jurnal Krisnadana*, 2(3), 412-423.
- Sari, N. K., Oktavianti, M. C., & Samsun. (2020). Analisis Karakter Segmen Abnormal pada Citra Mamografi dengan Menggunakan Berbagai Metode Preprocessing Citra. *Jurnal Ilmiah Giga*, 22, 1-8.
- Shah, E. (2024, January 5). *The Difference Between AI, ML and DL*. Retrieved from Scaler: <https://www.scaler.com/topics/artificial-intelligence-ml-dl/>
- Shin, D. J., & Kim, J. J. (2022). A Deep Learning Framework Performance Evaluation to Use YOLO in Nvidia Jetson Platform. *Applied Sciences*, 12(8), 1-19.
- Sindushree, D., Hedge, G. M., & Hedge, R. Y. (2020). In Vitro Evaluation of Botanicals and ITKs against Septoria lycopersici Causing Septoria Leaf Spot of Tomato. *International Journal of Current Microbiology and Applied Sciences*, 9(4), 407-415.
- Sitthitanasin, S., Korakngam, C., Kanhayart, T., & Kosticharoenkul, N. (2021). Characterization of Xanthomonas causing of Bacterial Leaf Spot of Tomato and Pepper in Thailand. 38, 80-89.
- Suwignyo, S., Hersanti, & Widiyanti, F. (2022). Pengaruh Kitosan Nano terhadap Penyakit Bercak Coklat (*Alternaria solani* Sor.) pada Tanaman Tomat. *Agrikultura*, 32(3), 239-247.
- Wagle, S., Harikrishnan, R., Ali, S., & Faseehuddin, M. (2022). Classification of Plant Leaves Using New Compact Convolutional Neural Network Models. *Plants*, 11, 24.
- Wati, C., Arsi, Karenina, T., Riyanto, & Nirwanto, Y. d. (2021). *Hama dan Penyakit Tanaman*. Bogor: Yayasan Kita Menulis.
- Watt, B. (2020). Early Blight on Tomato. *PlantwisePlus Knowledge Bank*.
- Wibowo, D. K. (2011). Program Aplikasi Gambe Battle Tank Menggunakan Bahasa Pemrograman Python.
- Wiguna, G., Sutarya, R., & Muliani, Y. (2015). Respon Beberapa Galur Tomat (*Lycopersicum Esculentum* Mill.) Terhadap Penyakit Busuk Daun (*Phytophthora Infestans* (Mont.) De Bary). *Mediagro*, 11(2), 1-10.
- Yan, Z., Wolters, A.-M., Jesús, N.-C., & Bai, Y. (2021). The Global Dimension of Tomato Yellow Leaf Curl Disease: Current Status and Breeding Perspectives. *Microorganism*, 9(4), 740.

LAMPIRAN

Lampiran 1 Coding pelatihan model dengan 3 lapisan konvolusi tanpa *Reduce Learning Rate* dan *Early Stopping*

```
from google.colab import drive
drive.mount ('/content/drive')
!ls "/content/drive/MyDrive/skripsi"

import TensorFlow as tf
from keras.preprocessing import image
from TensorFlow.keras.utils import img_to_array, array_to_img
from keras.optimizers import Adam
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Flatten, Dropout, Dense
from sklearn.model_selection import train_test_split
from keras.models import model_from_json
from keras.utils import to_categorical
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.image import imread
import cv2
import random
from TensorFlow.keras import models, layers, optimizers,
regularizers

dataset = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/skripsi/train',
    batch_size=32,
    image_size=(256, 256),
    shuffle=True
)

BATCH_SIZE = 16
IMAGE_SIZE = 256
CHANNELS=3
EPOCHS=50

class_names = dataset.class_names
```

```

class_names

#splitting dataset
def get_dataset_partitions_tf(ds, train_split=0.8,
val_split=0.1, test_split=0.1, shuffle=True,
shuffle_size=5000):
    assert (train_split + test_split + val_split) == 1

    ds_size = len(ds)

    if shuffle:
        ds = ds.shuffle(shuffle_size, seed=12)

    train_size = int(train_split * ds_size)
    val_size = int(val_split * ds_size)

    train_ds = ds.take(train_size)
    val_ds = ds.skip(train_size).take(val_size)
    test_ds = ds.skip(train_size).skip(val_size)

    return train_ds, val_ds, test_ds

train_ds, val_ds, test_ds = get_dataset_partitions_tf(dataset)

#preprocess
train_ds =
train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AU
TOTUNE)
val_ds =
val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTO
TUNE)
test_ds =
test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUT
OTUNE)
resize_and_rescale = tf.keras.Sequential([
    layers.experimental.preprocessing.Resizing(IMAGE_SIZE,
IMAGE_SIZE),
    layers.experimental.preprocessing.Rescaling(1.0/255),
])
input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 10

model = models.Sequential([

```

```

        layers.InputLayer(input_shape=(IMAGE_SIZE, IMAGE_SIZE,
CHANNELS)),
        resize_and_rescale,
        layers.Conv2D(16, kernel_size=(3,3), activation='relu'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(32, kernel_size=(3, 3), activation='relu'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation='relu'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),
        layers.Flatten(),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(n_classes, activation='softmax'),
    ])

model.build(input_shape=input_shape)

model.summary()

model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)

history = model.fit(
    train_ds,
    batch_size=BATCH_SIZE,
    validation_data=val_ds,
    verbose=1,
    epochs=50,
)

# grafik hasil latihan
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='accuracy')

```



```
plt.plot(history.history['val_accuracy'],
label='val_accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

Lampiran 2 Coding Pelatihan Model dengan Penambahan Lapisan Konvolusi dan Regularisasi

```
from google.colab import drive
drive.mount ('/content/drive')
!ls "/content/drive/MyDrive/skripsi"
```

```
#import library
import TensorFlow as tf
from keras.preprocessing import image
from TensorFlow.keras.utils import img_to_array, array_to_img
from keras.optimizers import Adam
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Flatten, Dropout, Dense
from sklearn.model_selection import train_test_split
from keras.models import model_from_json
from keras.utils import to_categorical
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.image import imread
import cv2
import random
from TensorFlow.keras import models, layers, optimizers,
regularizers
```

```

dataset = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/skripsi/train',
    batch_size=32,
    image_size=(256, 256),
    shuffle=True
)

BATCH_SIZE = 16
IMAGE_SIZE = 256
CHANNELS=3
EPOCHS=50

class_names = dataset.class_names
class_names

def get_dataset_partitions_tf(ds, train_split=0.8,
    val_split=0.1, test_split=0.1, shuffle=True,
    shuffle_size=5000):
    assert (train_split + test_split + val_split) == 1

    ds_size = len(ds)

    if shuffle:
        ds = ds.shuffle(shuffle_size, seed=12)

    train_size = int(train_split * ds_size)
    val_size = int(val_split * ds_size)

    train_ds = ds.take(train_size)
    val_ds = ds.skip(train_size).take(val_size)
    test_ds = ds.skip(train_size).skip(val_size)

    return train_ds, val_ds, test_ds

train_ds, val_ds, test_ds = get_dataset_partitions_tf(dataset)

train_ds =
train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
val_ds =
val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)

```

```

test_ds =
test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)

resize_and_rescale = tf.keras.Sequential([
    layers.experimental.preprocessing.Resizing(IMAGE_SIZE,
IMAGE_SIZE),
    layers.experimental.preprocessing.Rescaling(1.0/255),
])

input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 10

model = models.Sequential([
    layers.InputLayer(input_shape=(IMAGE_SIZE, IMAGE_SIZE,
CHANNELS)),
    resize_and_rescale,
    layers.Conv2D(16, kernel_size=(3,3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(32, kernel_size=(3, 3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, kernel_size=(3, 3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, kernel_size=(3, 3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu',
kernel_regularizer=regularizers.l2(0.01)),
    layers.Dropout(0.5),
    layers.Dense(n_classes, activation='softmax',
kernel_regularizer=regularizers.l2(0.01)),
])

model.build(input_shape=input_shape)

model.summary()

model.compile(
    optimizer='adam',

```

```

        loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
        metrics=['accuracy']
    )
    history = model.fit(
        train_ds,
        batch_size=BATCH_SIZE,
        validation_data=val_ds,
        verbose=1,
        epochs=50,
    )

    import matplotlib.pyplot as plt
    plt.figure(figsize=(12, 4))
    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'], label='accuracy')
    plt.plot(history.history['val_accuracy'],
             label='val_accuracy')
    plt.title('Training and Validation Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'], label='loss')
    plt.plot(history.history['val_loss'], label='val_loss')
    plt.title('Training and Validation Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()

    plt.show()

```

Lampiran 3 Coding Pelatihan Model Dengan Penambahan *Reduce Learning Rate*

```

from google.colab import drive
drive.mount ('/content/drive')
!ls "/content/drive/MyDrive/skripshi"

import TensorFlow as tf
from keras.preprocessing import image
from TensorFlow.keras.utils import img_to_array, array_to_img
from keras.optimizers import Adam

```

```

from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Flatten, Dropout, Dense
from sklearn.model_selection import train_test_split
from keras.models import model_from_json
from keras.utils import to_categorical
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.image import imread
import cv2
import random
from TensorFlow.keras import models, layers, optimizers,
regularizers
from TensorFlow.keras.callbacks import ReduceLROnPlateau
from TensorFlow.keras.callbacks import EarlyStopping,
ReduceLROnPlateau

dataset = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/skripshi/train',
    batch_size=32,
    image_size=(256, 256),
    shuffle=True
)

BATCH_SIZE = 16
IMAGE_SIZE = 256
CHANNELS=3
EPOCHS=80

class_names = dataset.class_names
class_names

def get_dataset_partitions_tf(ds, train_split=0.8,
val_split=0.1, test_split=0.1, shuffle=True,
shuffle_size=5000):
    assert (train_split + test_split + val_split) == 1

    ds_size = len(ds)

    if shuffle:
        ds = ds.shuffle(shuffle_size, seed=12)

    train_size = int(train_split * ds_size)
    val_size = int(val_split * ds_size)

    train_ds = ds.take(train_size)
    val_ds = ds.skip(train_size).take(val_size)
    test_ds = ds.skip(train_size).skip(val_size)

    return train_ds, val_ds, test_ds

```

```

train_ds, val_ds, test_ds = get_dataset_partitions_tf(dataset)

train_ds =
train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
val_ds =
val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
test_ds =
test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)

lr_reduction = ReduceLROnPlateau(monitor='val_loss',
                                  patience=3,
                                  verbose=1,
                                  factor=0.5,
                                  min_lr=0.00001)

early_stopping = EarlyStopping(monitor='val_loss',
                                patience=5,
                                restore_best_weights=True)

resize_and_rescale = tf.keras.Sequential([
    layers.experimental.preprocessing.Resizing(IMAGE_SIZE,
IMAGE_SIZE),
    layers.experimental.preprocessing.Rescaling(1.0/255),
])

input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 10

model = models.Sequential([
    layers.InputLayer(input_shape=(IMAGE_SIZE, IMAGE_SIZE,
CHANNELS)),
    resize_and_rescale,
    layers.Conv2D(16, kernel_size=(3,3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(32, kernel_size=(3, 3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, kernel_size=(3, 3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, kernel_size=(3, 3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu',
kernel_regularizer=regularizers.l2(0.01)),

```

```

        layers.Dropout(0.5),
        layers.Dense(n_classes, activation='softmax',
kernel_regularizer=regularizers.l2(0.01)),
    ])

model.build(input_shape=input_shape)

model.summary()

model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)

history = model.fit(
    train_ds,
    batch_size=BATCH_SIZE,
    validation_data=val_ds,
    verbose=1,
    epochs=80,
    callbacks=[lr_reduction, early_stopping]
)

plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'],
label='val_accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()

```

Lampiran 4 Coding Testing Model Terbaik

```

from google.colab import drive
drive.mount('/content/drive')
!ls '/content/drive/MyDrive/skripsi/thirdmodel.h5'

!pip install TensorFlow
import TensorFlow as tf
from TensorFlow.keras.models import load_model
import numpy as np
from TensorFlow.keras import models, layers, optimizers,
regularizers
from TensorFlow.keras.callbacks import ReduceLROnPlateau
from sklearn.metrics import precision_score

model_path = '/content/drive/MyDrive/skripsi/thirdmodel.h5'
model = load_model(model_path)

dataset = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/skripsi/train',
    batch_size=32,
    image_size=(256, 256),
    shuffle=True
)

class_names = dataset.class_names
class_names

def get_dataset_partitions_tf(ds, train_split=0.8,
val_split=0.1, test_split=0.1, shuffle=True,
shuffle_size=5000):
    assert (train_split + test_split + val_split) == 1

    ds_size = len(ds)

    if shuffle:
        ds = ds.shuffle(shuffle_size, seed=12)

    train_size = int(train_split * ds_size)
    val_size = int(val_split * ds_size)

    train_ds = ds.take(train_size)
    val_ds = ds.skip(train_size).take(val_size)

```



```

    test_ds = ds.skip(train_size).skip(val_size)

    return train_ds, val_ds, test_ds

train_ds, val_ds, test_ds = get_dataset_partitions_tf(dataset)

train_ds =
train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
val_ds =
val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
test_ds =
test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)

resize_and_rescale = tf.keras.Sequential([
    layers.experimental.preprocessing.Resizing(256, 256),
    layers.experimental.preprocessing.Rescaling(1.0/255),
])

lr_reduction = ReduceLROnPlateau(monitor='val_loss',
                                  patience=2,
                                  verbose=1,
                                  factor=0.5,
                                  min_lr=0.00001)

# Confusion Matrix train
y_true = []
y_pred = []

for images, labels in train_ds:
    predictions = model.predict(images)
    predicted_labels = np.argmax(predictions, axis=1)
    y_true.extend(labels)
    y_pred.extend(predicted_labels)

confusion_matrix = confusion_matrix(y_true, y_pred)

plt.figure(figsize=(7, 5))
sns.heatmap(confusion_matrix, annot=True, fmt="d",
            cmap="Blues", xticklabels=class_names,
            yticklabels=class_names)
plt.xlabel("Predicted Label")

```

```

plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()

# Confusion Matrix test
y_true = []
y_pred = []

for images, labels in test_ds:
    predictions = model.predict(images)
    predicted_labels = np.argmax(predictions, axis=1)
    y_true.extend(labels)
    y_pred.extend(predicted_labels)

confusion_matrix = confusion_matrix(y_true, y_pred)

plt.figure(figsize=(7, 5))
sns.heatmap(confusion_matrix, annot=True, fmt="d",
            cmap="Blues", xticklabels=class_names,
            yticklabels=class_names)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()

# Display Mismatch
for class_idx in range(len(class_names)):
    misclassified_indices = np.where((np.array(y_true) ==
class_idx) & (np.array(y_pred) != class_idx))[0]

    num_images_to_display = min(5, len(misclassified_indices))
    if num_images_to_display > 0:
        print(f"Misclassified images for class:
{class_names[class_idx]}")
        plt.figure(figsize=(15, 5))
        for i in range(num_images_to_display):
            idx = misclassified_indices[i]
            image, label =
list(test_ds.unbatch().skip(idx).take(1))[0]
            predicted_label = y_pred[idx]
            plt.subplot(1, num_images_to_display, i + 1)
            plt.imshow(image.numpy().astype("uint8"))

```

```
plt.title(f"True: {class_names[label.numpy()]},
Predicted: {class_names[predicted_label]}")
plt.axis("off")
plt.show()
```

Lampiran 5 Coding User Interface Aplikasi

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/main"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="@drawable/background_img"
tools:context=".MainActivity">

    <TextView
        android:id="@+id/appname"
        android:layout_width="match_parent"
        android:layout_height="30dp"
        android:layout_marginTop="20dp"
        android:fontFamily="@font/poppins"
        android:text="Potato Disease Identification App"
        android:textAlignment="center"
        android:textColor="#FFFFFF"
        android:textSize="20dp"
        android:textStyle="bold" />

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="350dp"
        android:layout_margin="25dp"
        android:id="@+id/imageView"
        android:layout_below="@+id/appname"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/linLayout"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/imageView"
        android:orientation="horizontal"
        android:gravity="center"
        >

        <Button
            android:id="@+id/selectBtn"
            android:layout_width="125dp"
            android:layout_height="wrap_content"
```

```

        android:layout_marginRight="30dp"
        android:background="@drawable/custom_button1"
        android:fontFamily="@font/poppins"
        android:layout_margin="10dp"
        android:text="Pilih Gambar"
        android:textAllCaps="false"
        android:typeface="normal" />

<Button
    android:id="@+id/identifyBtn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:width="125dp"
    android:background="@drawable/custom_button2"
    android:fontFamily="@font/poppins"
    android:text="Identifikasi"
    android:textAllCaps="false"
    android:typeface="normal" />
</LinearLayout>

<TextView
    android:id="@+id/restext"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/linLayout"
    android:layout_marginLeft="40dp"
    android:layout_marginTop="21dp"
    android:text="Hasil Identifikasi : "
    android:textColor="#FFFFFF"
    android:textSize="18dp"
    android:textStyle="bold" />

<TextView
    android:id="@+id/resView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/linLayout"
    android:layout_marginLeft="40dp"
    android:layout_marginTop="57dp"
    android:textColor="#FFFFFF"
    android:textSize="18dp"
    android:textStyle="bold" />

</RelativeLayout>

```

Lampiran 6 Coding Kotlin Untuk Fitur Aplikasi

```
package com.example.skripsitest

import android.content.Intent
import android.graphics.Bitmap
import android.graphics.Color
import android.net.Uri
import android.os.Bundle
import android.provider.MediaStore
import android.widget.Button
import android.widget.ImageView
import android.widget.TextView
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import com.example.skripsitest.ml.Model2Dauntomat
import org.tensorflow.lite.DataType
import org.tensorflow.lite.support.common.ops.NormalizeOp
import org.tensorflow.lite.support.image.ImageProcessor
import org.tensorflow.lite.support.image.TensorImage
import org.tensorflow.lite.support.image.ops.ResizeOp
import org.tensorflow.lite.support.tensorbuffer.TensorBuffer

class MainActivity : AppCompatActivity() {

    private lateinit var selectBtn: Button
    private lateinit var identifyBtn: Button
    private lateinit var resView: TextView
    private lateinit var imageView: ImageView
    private lateinit var bitmap: Bitmap

    // Preprocess image
    private val imageProcessor = ImageProcessor.Builder()
        .add(ResizeOp(256, 256, ResizeOp.ResizeMethod.BILINEAR))
        .add(NormalizeOp(0.0f, 1.0f / 255.0f))
        .build()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Initialize UI elements
        selectBtn = findViewById(R.id.selectBtn)
        identifyBtn = findViewById(R.id.identifyBtn)
        resView = findViewById(R.id.resView)
        imageView = findViewById(R.id.imageView)

        val labels =
            application.assets.open("labels.txt").bufferedReader().readLines()

        // Set padding for insets

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main))
```

```

{ v, insets ->
    val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
    v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom)
    insets
}

// Setup listener for selectBtn
selectBtn.setOnClickListener {
    val intent = Intent()
    intent.action = Intent.ACTION_GET_CONTENT
    intent.type = "image/*"
    startActivityForResult(intent, 100)
}

// Setup listener for predictBtn
identifyBtn.setOnClickListener {
    if (::bitmap.isInitialized) {
        predictImage(labels)
    } else {
        resView.text = "Please select an image first."
    }
}
}

fun preprocessImage(bitmap: Bitmap): TensorImage {
    // Convert the Bitmap to a TensorImage
    val tensorImage = TensorImage()
    tensorImage.load(bitmap)

    // Expand the dimensions to match the expected input shape
of the model
    val expandedTensorImage = TensorImage(DataType.FLOAT32)
    expandedTensorImage.load(bitmap)
    expandedTensorImage.buffer.rewind() // Rewind the buffer
to the beginning
    return expandedTensorImage
}

private fun predictImage(labels: List<String>) {
    val tensorImage = preprocessImage(bitmap)

    val model = Model2Dauntomat.newInstance(this)

    val inputFeature0 =
TensorBuffer.createFixedSize(intArrayOf(1, 256, 256, 3),
DataType.FLOAT32)
    inputFeature0.loadBuffer(tensorImage.buffer)

    val outputs = model.process(inputFeature0)
    val outputFeature0 =
outputs.outputFeature0AsTensorBuffer.floatArray

```

```

        var maxIdx = 0
        outputFeature0.forEachIndexed { index, fl ->
            if (outputFeature0[maxIdx]<fl) {
                maxIdx = index
            }
        }

        resView.setText(labels[maxIdx])

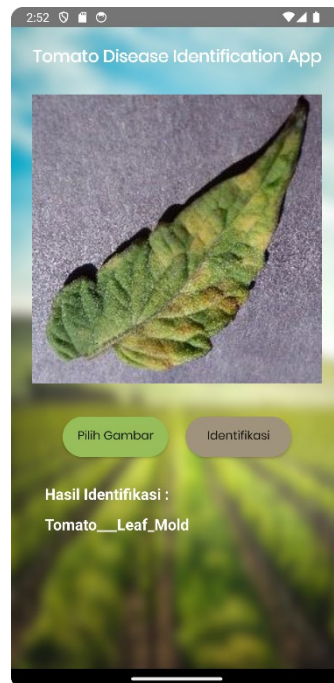
        model.close()
    }

    override fun onActivityResult(requestCode: Int, resultCode:
Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)

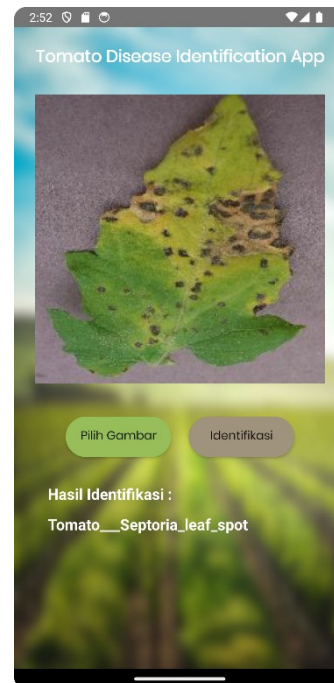
        if (requestCode == 100 && resultCode == RESULT_OK && data
!= null) {
            val uri: Uri? = data.data
            if (uri != null) {
                bitmap =
MediaStore.Images.Media.getBitmap(this.contentResolver, uri)
                imageView.setImageBitmap(bitmap)
            }
        }
    }
}

```

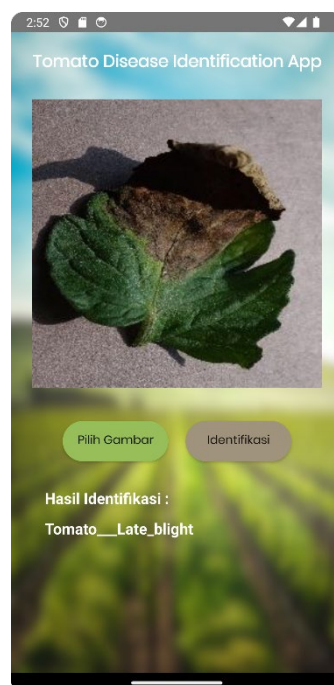
Lampiran 7 Tampilan Aplikasi Pada Sistem Operasi Android



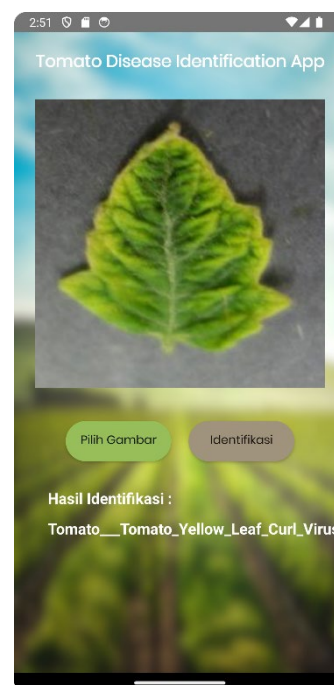
(a)



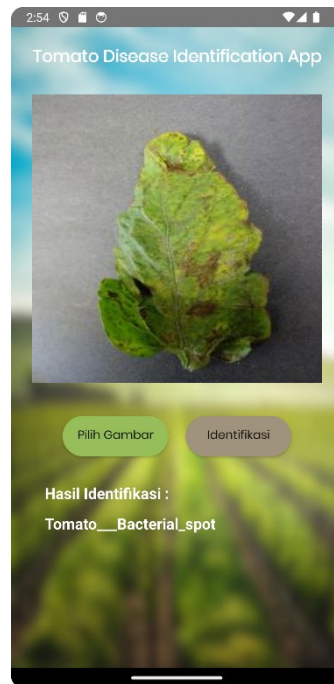
(b)



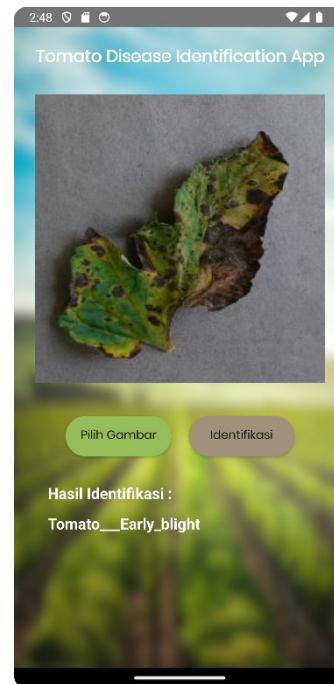
(c)



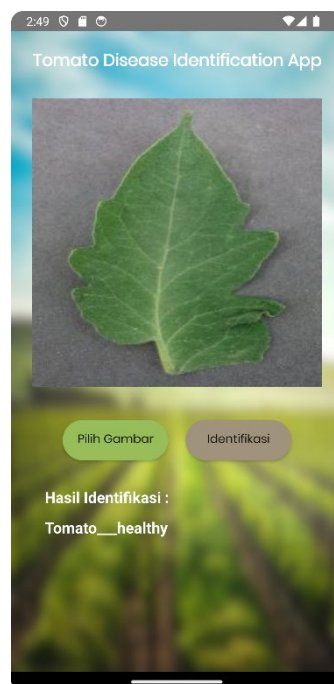
(d)



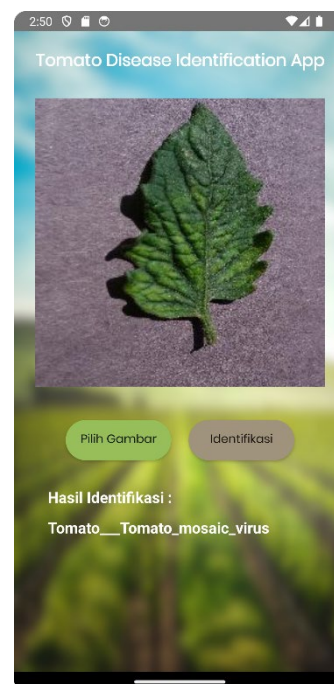
(e)



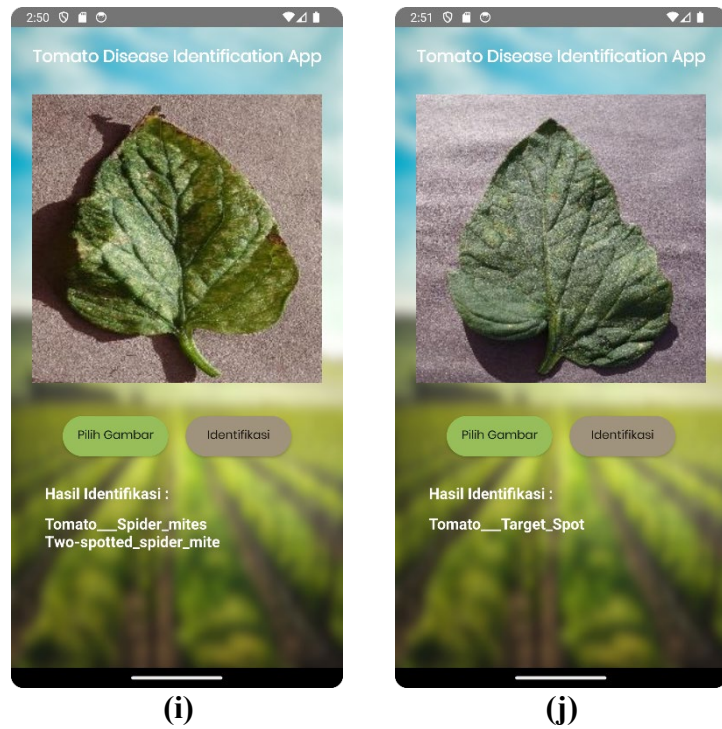
(f)



(g)



(h)



Gambar hasil identifikasi aplikasi tomat dalam Sistem Operasi android pada penyakit (a) *leafmold*, (b) *septoria leaf spot*, (c) *late blight*, (d) *yellow leaf curl virus*, (e) *bacterial spot*, (f) *early blight*, (g) *healthy*, (h) *mosaic virus*, (i) *two spotted spider mites*, dan (j) *target spot*