BAB II
KAJIAN PUSTAKA

2.1 Teori-teori terkait bahasan penelitian (Relevan Theories)

Teori- teori penunjang yang relevan dengan penelitian dan sesuai dengan pembahasan penelitian meliputi:

2.1.1 Rancang Bangun

Rancang bangun merupakan tahap penting dalam pengembangan sistem yang menampilkan hasil analisis kebutuhan menjadi kerangka kerja yang siap implementasi. Tahapan ini mencakup pembuatan model sistem, organisasi basis data, serta perancangan desain antarmuka pengguna untuk memastikan sistem berjalan semestinya. Proses ini tidak hanya ditujukan untuk membangun sistem baru, tetapi juga untuk mengubah kinerja sistem yang sudah ada (Syahputra et al., 2024). Perancangan sistem melibatkan pendekatan sistematis dalam membangun sebuah sistem, mulai dari pengorganisasian struktur data hingga pengujian desain sebelum sistem diimplementasikan. Tahap ini sangat krusial untuk memastikan bahwa sistem yang dihasilkan mampu memenuhi kebutuhan pengguna (Prasetya & Sujatmiko, 2022).

Berdasarkan dua penjelasan di atas, rancang bangun merupakan suatu proses yang menggambarkan, merencanakan, dan merancang, kemudian menggabungkannya menjadi sistem yang berfungsi secara optimal. Proses ini mencakup konfigurasi yang kompleks yang bertujuan untuk merancang sistem baru atau menyempurnakan sistem yang sudah ada, sehingga sistem tersebut dapat beroperasi sesuai dengan tujuan yang telah disepakati.

2.1.2 Aplikasi

Aplikasi merupakan perangkat lunak dirancang untuk menjalankan fungsi tertentu yang mengubah data mentah menjadi informasi yang ditujukan

bagi pengguna, pada praktik umumnya dengan cara mengubah metode manual tradisional menjadi alur kerja yang terotomatisasi dan efisien. Pengembangan aplikasi memerlukan penulisan kode instruksi rinci untuk mengelola masukan data, perubahannya, serta hasil yang dikeluarkan (Prasetya & Sujatmiko, 2022). Aplikasi perangkat lunak dapat didefinisikan sebagai alat yang dirancang untuk memungkinkan individu memanfaatkan kemampuan komputer melalui program yang mudah diakses, yang terdiri dari pemrosesan informasi sesuai instruksi pemrograman tertentu untuk menghasilkan keluaran yang fungsional. Tindakan ini mencakup metode yang dilakukan secara manual menjadi terkomputerisasi, termasuk merancang strategi, membangun sistem, dan memberikan dukungan setelah implementasi untuk memastikan hasil yang berfungsi dengan baik (Surahmat, 2023).

Berdasarkan dua penjelasan sebelumnya, aplikasi adalah paket perangkat lunak yang terdiri dari perintah-perintah terprogram untuk mengelola data secara terorganisir. Aplikasi menggabungkan berbagai komponen seperti algoritma, antarmuka, dan basis data untuk membentuk sistem yang berfungsi dengan baik. Metode ini tidak hanya menciptakan program baru yang inovatif, tetapi juga memperbarui sistem lama dengan perencanaan yang baik, sehingga sistem tersebut dapat berfungsi memenuhi kebutuhan dan tujuan saat ini.

2.1.3 Klasifikasi

Klasifikasi merupakan proses penting dalam pengolahan data dan pengembangan sistem yang bertujuan untuk mengelompokkan objek, data, atau informasi ke dalam kategori tertentu berdasarkan karakteristik yang dimiliki. Proses ini dilakukan dengan menggunakan aturan atau kriteria tertentu agar setiap data dapat ditempatkan secara sistematis dan mudah dianalisis. Dalam konteks sistem informasi, klasifikasi berperan dalam menyederhanakan kompleksitas data sehingga mempermudah proses pencarian, pengolahan, serta pengambilan keputusan (Prasetya & Sujatmiko,

2022). Klasifikasi juga mencakup metode analitis yang digunakan untuk memetakan data ke dalam kelas yang telah ditentukan sebelumnya, baik secara manual maupun menggunakan teknik komputasi seperti *machine learning*. Proses ini biasanya melibatkan tahap identifikasi fitur, pemilihan model, serta evaluasi hasil untuk memastikan tingkat akurasi yang optimal. Dengan adanya klasifikasi, sistem dapat bekerja lebih efisien dalam mengenali pola dan memberikan prediksi berdasarkan data yang tersedia (Firmansyah et al., 2021).

Berdasarkan penjelasan tersebut, klasifikasi merupakan suatu proses sistematis dalam mengelompokkan data atau objek ke dalam kategori tertentu berdasarkan kesamaan karakteristik yang dimiliki. Proses ini bertujuan untuk mempermudah pengelolaan data, meningkatkan efisiensi analisis, serta mendukung pengambilan keputusan yang lebih tepat. Dalam implementasinya, klasifikasi tidak hanya digunakan untuk mengorganisasi data baru, tetapi juga untuk menyempurnakan struktur data yang sudah ada agar lebih terarah dan sesuai dengan kebutuhan pengguna.

2.1.4 Penyakit pada Daun Tomat

1. Bercak Daun

   Gejala ini terjadi pada tanaman tomat karena jamur atau bakteri menghisap nutrisi dan merusak jaringan di permukaan bawah daun. Tanda serangan adalah bercak cokelat kecil berbentuk cincin konsentris dengan halo kuning. Serangan yang kuat dapat mengakibatkan daun menguning dan gugur, menghambat pertumbuhan tanaman, dan bahkan batang menjadi layu, setelah daun berubah warna menjadi bintik kehitaman, dan keriput (Hartati et al., 2024).

Gambar 2. 1 *Leaf Spot* (bercak daun)

2. Busuk Daun

Gejala ini terjadi pada tanaman tomat karena jamur *Phytophthora infestans* merusak jaringan di permukaan bawah daun. Tanda serangan adalah bercak basah kehitaman dengan lapisan putih. Serangan yang kuat dapat mengakibatkan daun menguning dan gugur, menghambat tanaman untuk tumbuh, dan bahkan batang tanaman menjadi layu, setelah daun berubah warna menjadi bintik coklat, dan keriput (Abadi et al., 2024).

Gambar 2. 2 *Leaf Blight* (busuk daun)

3. Jamur Daun

Gejala ini terjadi pada tanaman tomat karena jamur *Oidium neolycopersici* membentuk lapisan spora di permukaan atas daun. Tanda serangan adalah lapisan tepung putih keabu-abuan seperti bedak. Serangan yang kuat dapat mengakibatkan daun menguning dan keriput, menghambat tanaman untuk tumbuh, dan bahkan pucuk tanaman menjadi mati, setelah daun berubah warna menjadi kecokelatan, dan gugur (Istifadah & Maharani, 2023).

Gambar 2. 3 *Powdery Mildew* (jamur daun)

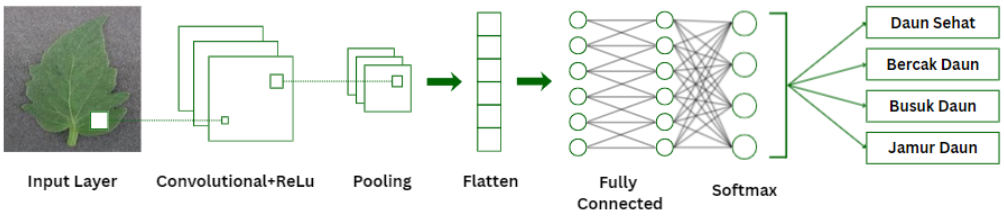
2.1.5 Algoritma

Algoritma merupakan komponen penting dalam pengembangan sistem yang berfungsi sebagai serangkaian langkah logis dan terstruktur untuk menyelesaikan suatu permasalahan. Algoritma mengubah kebutuhan atau permasalahan yang telah dianalisis menjadi prosedur sistematis yang dapat diimplementasikan dalam bentuk program komputer (Paraijun et al., 2022). Algoritma juga mencakup pendekatan sistematis dalam menyelesaikan masalah, dimulai dari identifikasi *input*, proses pengolahan, hingga menghasilkan *output* yang diinginkan. Dalam penerapannya, algoritma harus memenuhi kriteria tertentu seperti kejelasan langkah, keterbatasan proses, serta efisiensi waktu dan penggunaan sumber daya (Abdiansah et al., 2025).

Berdasarkan dua penjelasan di atas, algoritma merupakan suatu proses yang menggambarkan, merencanakan, dan menyusun langkah-langkah logis secara sistematis untuk menyelesaikan suatu masalah, kemudian mengimplementasikannya ke dalam sistem yang berfungsi secara optimal. Proses ini mencakup struktur yang terorganisir yang bertujuan untuk membangun solusi baru atau menyempurnakan proses yang sudah ada, sehingga sistem dapat beroperasi secara efektif sesuai dengan tujuan yang telah ditetapkan.

2.1.6 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) adalah jenis arsitektur *deep learning* yang dirancang khusus untuk memproses data citra melalui ekstraksi fitur secara otomatis. CNN didefinisikan sebagai jaringan saraf yang meniru cara kerja visual cortex manusia, dengan lapisan utama seperti *convolutional layer* yang berfungsi untuk mendeteksi pola seperti tepi dan tekstur, *pooling layer* untuk melakukan reduksi dimensi, serta *Fully connected layer* yang digunakan dalam proses klasifikasi akhir (Salauddin et al., 2025). Arsitektur CNN secara umum terdiri dari dua bagian utama, yaitu *feature learning* dan *classification*, di mana pada tahap *feature learning* jaringan melakukan ekstraksi fitur dari citra *input* melalui serangkaian proses konvolusi dan *Pooling*, sedangkan pada tahap *classification* fitur yang telah diperoleh akan diproses lebih lanjut menggunakan *Fully connected layer* untuk menghasilkan *output* berupa kelas atau kategori tertentu (Prayoga et al., 2023), sebagaimana ditunjukkan pada Gambar 3 dibawah.

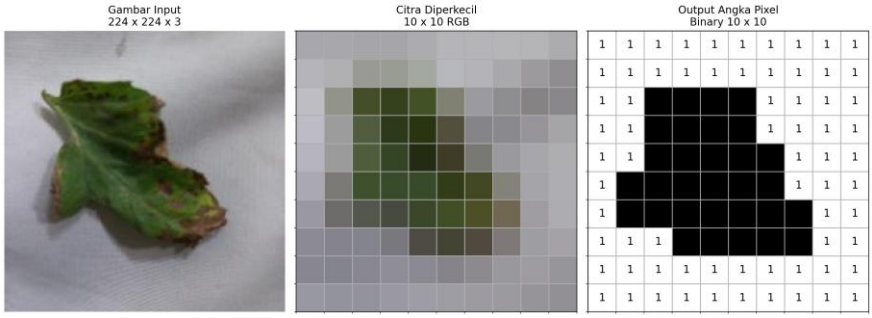


Gambar 2. 4 Arsitektur CNN

Berikut merupakan penjelasan dari bagian arsitektur CNN sebagai berikut:

a. *Input Layer*

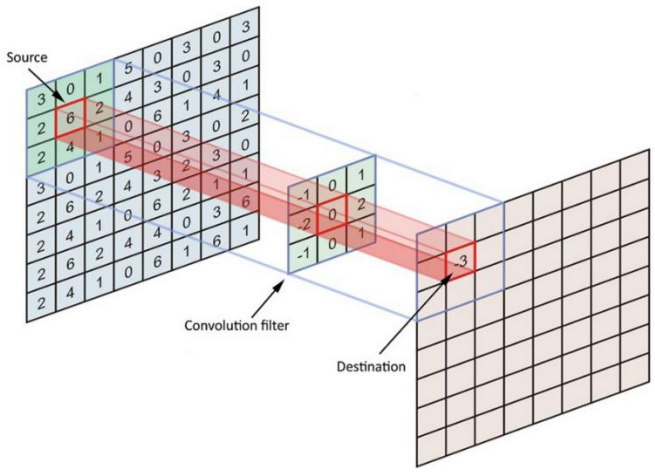
Input layer berfungsi sebagai pintu masuk data citra mentah, biasanya berukuran tetap seperti 224x224x3 piksel untuk gambar *RGB*. Lapisan ini menyiapkan *tensor input* untuk pemrosesan selanjutnya tanpa transformasi, memastikan kompatibilitas dengan lapisan konvolusi (Yulita et al., 2023).



Gambar 2. 5 Contoh *input layer* (Yulita et al., 2023)

b. *Convolutional Layer*

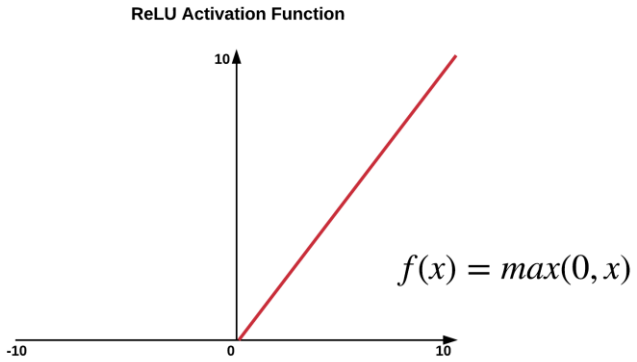
Convolutional Layer melakukan operasi konvolusi menggunakan *filter* atau *kernel* untuk mengekstrak fitur lokal seperti tepi dan tekstur dari *input* gambar. Setiap *filter* menghasilkan *feature map* yang menangkap pola spesifik, mengurangi dimensi spasial sambil mempertahankan informasi penting (Yulita et al., 2023).



Gambar 2. 6 Proses dari *Convolutional Layer* (Yulita et al., 2023)

c. *Activation Layer*

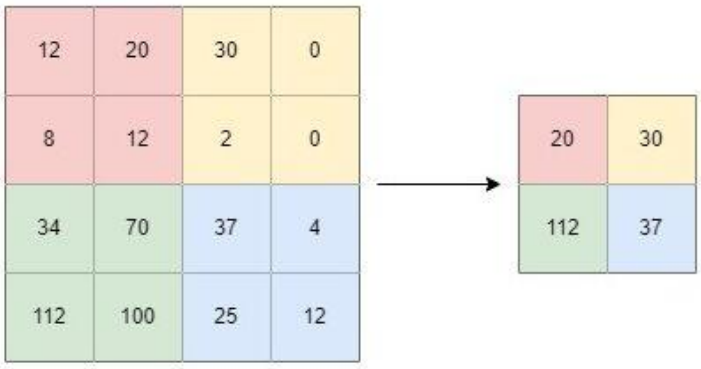
Activation Layer menerapkan fungsi non-linear seperti ReLU untuk menambahkan kemampuan non-linearitas pada *feature map* hasil konvolusi. Fungsi ini mengubah nilai negatif menjadi nol, sehingga mempercepat konvergensi dan mengurangi vanishing gradient (Yulita et al., 2023).



Gambar 2. 7 Proses Aktivasi ReLU (Yulita et al., 2023)

d. *Pooling Layer*

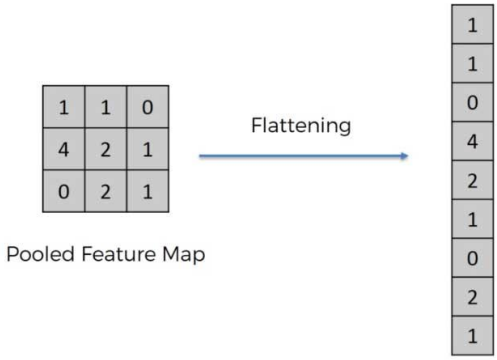
Pooling layer melakukan subsampling dengan mengambil statistik ringkasan seperti maksimum (*max pooling*) atau rata-rata (average *pooling*) dari wilayah *feature map*. Lapisan ini mengurangi dimensi spasial, menurunkan jumlah parameter, dan memberikan invariansi translasi untuk fitur yang lebih salient (Yulita et al., 2023).



Gambar 2. 8 Operasi *Max Polling* (Yulita et al., 2023)

e. *Flatten layer*

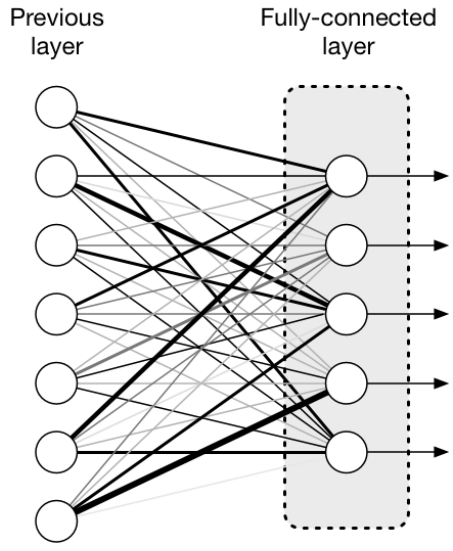
Flatten layer mengonversi seluruh *feature map* multi-dimensi dari lapisan sebelumnya menjadi vektor satu dimensi. Proses ini mempersiapkan data untuk lapisan *fully connected* dengan meratakan matriks menjadi *input* linier (Yulita et al., 2023).



Gambar 2. 9 *Flatten layer* (Yulita et al., 2023)

f. *Fully connected layer*

Fully connected layer menghubungkan setiap neuron di lapisan ini dengan semua neuron dari lapisan sebelumnya untuk melakukan klasifikasi atau regresi tingkat tinggi. Lapisan ini mengintegrasikan fitur global dan menerapkan bobot penuh untuk keputusan akhir (Yulita et al., 2023).

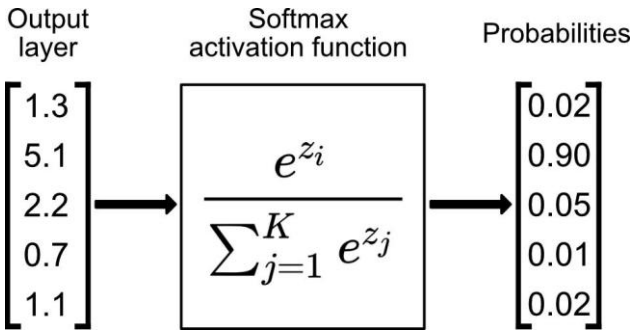


Gambar 2. 10 Proses *Fully connected layer* (Yulita et al., 2023)

g. Fungsi Aktivasi*: Softmax*

Fungsi Aktivasi *Softmax* mengubah *output* raw dari lapisan *fully connected* menjadi distribusi probabilitas yang dijumlahkan menjadi 1

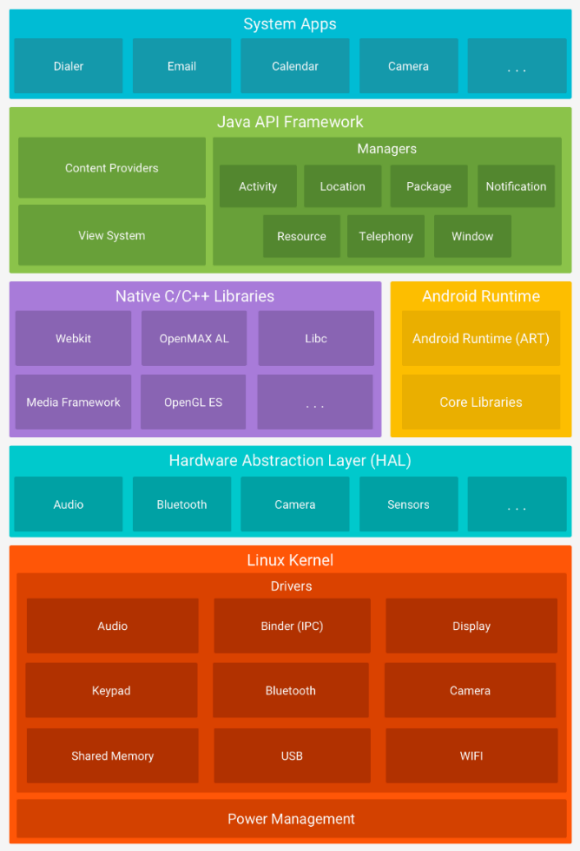
untuk setiap kelas. Fungsi ini ideal untuk klasifikasi multi-kelas karena menyoroti kelas dengan probabilitas tertinggi (Yulita et al., 2023).



Gambar 2. 11 Fungsi *Aktivasi Softmax* (Yulita et al., 2023)

2.1.7 Android

Android merupakan sistem operasi berbasis *open source* yang dirancang untuk mendukung berbagai perangkat mobile seperti smartphone, tablet, dan perangkat pintar lainnya. Android dikembangkan oleh Google dengan tujuan menyediakan platform yang fleksibel dan mudah digunakan bagi pengembang dalam membangun aplikasi yang interaktif dan fungsional. Sistem operasi ini berbasis *kernel* Linux yang memungkinkan pengelolaan perangkat keras, manajemen memori, serta keamanan sistem berjalan secara optimal. Selain itu, Android menyediakan *Software Development Kit* (SDK) yang lengkap, sehingga memudahkan pengembang dalam merancang, menguji, dan mengimplementasikan aplikasi (Marchiori et al., 2022).



Gambar 2. 12 Arsitektur Android (Sumber: https://developer.android.com/guide/platform)

Berikut merupakan penjelasan dari bagian arsitektur Android sebagai berikut:

a. *Linux Kernel*

Linux *Kernel* merupakan lapisan paling dasar yang berfungsi sebagai inti sistem operasi Android. Lapisan ini bertanggung jawab dalam mengelola perangkat keras seperti driver kamera, audio, WiFi, layar, serta manajemen daya dan memori (Marchiori et al., 2022)

b. *Libraries*

Libraries merupakan kumpulan pustaka yang menyediakan berbagai fungsi penting untuk mendukung kinerja sistem Android. Libraries membantu pengembang dalam memanfaatkan fungsi tingkat

rendah tanpa harus membangun semuanya dari awal (Marchiori et al., 2022).

c. *Android Runtime*

Android Runtime adalah lapisan yang menyediakan lingkungan eksekusi bagi aplikasi Android. Lapisan ini memungkinkan aplikasi berjalan secara efisien dengan manajemen memori yang baik serta mendukung eksekusi banyak aplikasi secara bersamaan (Marchiori et al., 2022).

d. *Application Framework*

Application *Frame*work merupakan lapisan yang menyediakan layanan dan API bagi pengembang untuk membangun aplikasi. Komponen di dalamnya meliputi *Activity* Manager, Window Manager, Content Providers, View System, dan berbagai manager lainnya seperti *Notification* dan Resource Manager (Marchiori et al., 2022).

e. *Applications*

Applications merupakan lapisan paling atas yang berisi aplikasi yang digunakan langsung oleh pengguna, seperti Home, Contacts, Phone, *Browser*, dan aplikasi lainnya. Lapisan ini berinteraksi dengan Application *Frame*work untuk mengakses layanan sistem dan menjalankan fungsi sesuai kebutuhan pengguna (Marchiori et al., 2022).

2.1.8 MobileNetV2

MobileNetV2 merupakan salah satu arsitektur *Convolutional Neural Network* (CNN) yang berperan penting dalam pengembangan sistem *computer vision*, khususnya pada perangkat dengan keterbatasan sumber daya seperti ponsel pintar dan perangkat *edge*. Arsitektur ini mengubah kebutuhan pemrosesan citra menjadi model yang efisien dengan tetap mempertahankan tingkat akurasi yang tinggi (Nguyen et al., 2023). MobileNetV2 juga mencakup pendekatan sistematis dalam pengolahan data citra, dimulai dari tahap ekstraksi

fitur menggunakan operasi konvolusi ringan hingga proses klasifikasi yang efisien. Dalam implementasinya, arsitektur ini dirancang untuk meminimalkan kompleksitas perhitungan tanpa mengorbankan kualitas hasil prediksi, sehingga sangat sesuai digunakan pada lingkungan dengan keterbatasan perangkat keras (Yulita et al., 2023).

Berdasarkan penjelasan tersebut, MobileNetV2 merupakan suatu arsitektur jaringan saraf yang dirancang secara sistematis untuk mengoptimalkan proses pengolahan citra dengan memanfaatkan struktur yang efisien, kemudian mengimplementasikannya ke dalam sistem yang mampu berjalan secara optimal pada perangkat dengan sumber daya terbatas. Proses ini mencakup konfigurasi yang terstruktur untuk membangun model yang ringan atau menyempurnakan arsitektur sebelumnya, sehingga sistem dapat beroperasi secara cepat, hemat memori, dan tetap akurat sesuai dengan tujuan yang telah ditetapkan.

2.1.9 Python

Python merupakan bahasa pemrograman serbaguna yang didukung pustaka standar komprehensif untuk pengembangan aplikasi web, analisis data, *machine learning*, serta otomatisasi tugas, dengan interpreter yang memfasilitasi pengujian cepat dan portabilitas lintas platform seperti Windows, Linux, dan macOS (Fahmi, 2023).

Gambar 2. 13 Python (Sumber : https://en.m.wikipedia.org/)

2.1.10 Kotlin

Kotlin merupakan bahasa pemrograman modern yang dirancang untuk mendukung pengembangan aplikasi secara efisien dan produktif, khususnya pada platform Android, serta dapat digunakan untuk pengembangan aplikasi web, backend, dan multiplatform. Kotlin dikembangkan oleh JetBrains dengan tujuan meningkatkan keamanan, keterbacaan kode, dan interoperabilitas dengan bahasa Java, sehingga memungkinkan pengembang untuk memanfaatkan ekosistem Java yang sudah ada dengan lebih optimal (Sanjaya & Susilo, 2024).

Gambar 2. 14 Kotlin (Sumber : https://en.m.wikipedia.org/)

2.1.11 TensorFlow

TensorFlow merupakan *frame*work pemrograman *open-source* untuk *machine learning* dan *deep learning* yang dikembangkan oleh Google Brain Team, didesain untuk membangun, melatih, dan menyebarkan model kecerdasan buatan mulai dari percobaan penelitian hingga skala produksi besar. *Framework* ini menyediakan kumpulan pustaka dan alat yang kaya untuk membangun jaringan saraf tiruan serta mendukung eksekusi lintas platform (Nyoman Tri Anindia Putra et al., 2023).

2.1.12 TensorFlow Lite

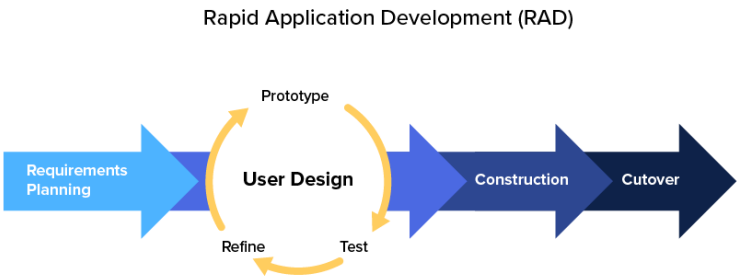
TensorFlow Lite merupakan versi ringan dan efisien dari *framework* TensorFlow yang dirancang khusus untuk menjalankan model *machine learning* pada perangkat dengan sumber daya terbatas, seperti ponsel pintar, tablet, dan IoT, dengan fokus pada inferensi yang cepat dan

latensi rendah. Library ini memungkinkan model berjalan langsung di perangkat tanpa bergantung pada koneksi internet sehingga mempercepat respon (Nyoman Tri Anindia Putra et al., 2023).

2.1.13 *Rapid Application Development* (RAD)

Rapid Application Development (RAD) merupakan salah satu metode pengembangan sistem yang menekankan pada kecepatan proses pembangunan aplikasi melalui pendekatan iteratif, keterlibatan aktif pengguna, serta penggunaan *prototyping* secara berkelanjutan. Metode ini dirancang untuk menghasilkan sistem dalam waktu relatif singkat dengan tetap memperhatikan kualitas dan kesesuaian kebutuhan pengguna. RAD memungkinkan adanya komunikasi yang intens antara pengembang dan pengguna, sehingga setiap perubahan kebutuhan dapat segera diakomodasi selama proses pengembangan berlangsung (Simanungkalit et al., 2023). RAD mencakup pendekatan sistematis dalam menciptakan sistem, dimulai dari perencanaan kebutuhan, proses perancangan bersama pengguna, pembangunan sistem secara cepat, hingga tahap implementasi. Dalam prosesnya, metode ini memungkinkan adanya umpan balik secara langsung dari pengguna, sehingga setiap kekurangan atau ketidaksesuaian dapat segera diperbaiki. Hal ini menjadikan RAD sebagai metode yang fleksibel dan adaptif terhadap perubahan kebutuhan selama proses pengembangan berlangsung (Nurman Hidayat & Kusuma Hati, 2021).

Secara teoritis, metode *Rapid Application Development* (RAD) terdiri dari empat tahapan utama yang saling terintegrasi, yaitu sebagai berikut:



Gambar 2. 15 Tahapan Metode RAD (Simanungkalit et al., 2023)

1. *Requirements Planning* (Perencanaan Kebutuhan)

Tahap ini merupakan fase awal yang bertujuan untuk mengidentifikasi kebutuhan sistem dan tujuan pengembangan aplikasi. Pada tahap ini, analis dan pengguna bekerja sama untuk mendefinisikan ruang lingkup sistem, kebutuhan fungsional maupun non-fungsional, serta batasan yang ada. Proses ini sangat penting untuk memastikan bahwa sistem yang dikembangkan sesuai dengan kebutuhan pengguna dan dapat menyelesaikan permasalahan yang dihadapi (Simanungkalit et al., 2023).

2. *User Design* (Perancangan Bersama Pengguna)

Pada tahap ini dilakukan proses perancangan sistem secara interaktif dengan melibatkan pengguna secara langsung. Pengembang membuat desain awal berupa prototipe, model sistem, serta diagram seperti UML (use case, activity, *class*, dan *Sequence* diagram). Pengguna kemudian memberikan umpan balik terhadap desain tersebut, sehingga terjadi proses evaluasi dan penyempurnaan secara berulang hingga diperoleh rancangan yang sesuai dengan kebutuhan operasional (Simanungkalit et al., 2023).

3. *Construction* (Konstruksi)

Tahap konstruksi merupakan fase pengembangan sistem secara nyata berdasarkan desain yang telah disepakati. Pada tahap ini dilakukan proses pengkodean, pembangunan basis data, serta integrasi komponen sistem. Selain itu, dilakukan juga pengujian secara bertahap untuk memastikan bahwa setiap fungsi sistem berjalan dengan baik. Keunggulan RAD pada tahap ini adalah penggunaan komponen yang dapat digunakan kembali (*reusable components*) sehingga mempercepat proses pembangunan aplikasi (Simanungkalit et al., 2023).

4. *Cutover* (Implementasi dan Penerapan)

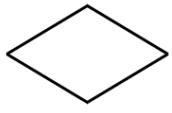
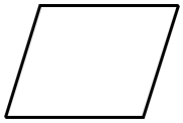
Tahap terakhir adalah implementasi sistem ke lingkungan pengguna. Pada fase ini dilakukan pengujian akhir, pelatihan pengguna, serta proses migrasi data jika diperlukan. Sistem yang telah dikembangkan kemudian dijalankan secara penuh dan dievaluasi untuk memastikan bahwa sistem telah memenuhi kebutuhan pengguna. Tahap ini juga mencakup pemeliharaan awal untuk menjamin sistem dapat beroperasi secara optimal (Simanungkalit et al., 2023).

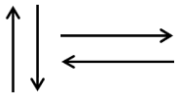
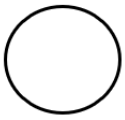
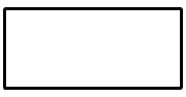
2.1.14 *Tools* Perancangan

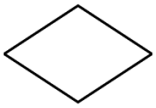
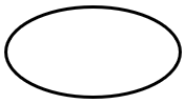
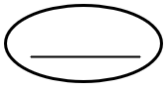
1. Flowchart

Flowchart merupakan salah satu alat bantu visual yang digunakan untuk menggambarkan alur proses atau langkah-langkah dalam suatu sistem secara sistematis dan terstruktur. *Flowchart* menyajikan urutan proses dalam bentuk simbol-simbol grafis seperti terminator (mulai/selesai), process (proses), *input*/*output*, decision (percabangan), dan flowline (alur), sehingga memudahkan pemahaman terhadap logika kerja suatu sistem atau algoritma (Zainab Tuasamu et al., 2023).

Tabel 2. 1 Simbol Flowchart (Sumber : https://app.diagrams.net/)

Simbol	Nama	Fungsi
	*Terminal*	Digunakan untuk menggambarkan awal dan akhir dari suatu kegiatan.
	*Decision*	Digunakan untuk menggambarkan proses pengujian suatu kondisi yang ada.
	*Input/Output*	Digunakan untuk menggambarkan proses masukan data

		yang berupa pembacaan data dan sekaligus proses keluaran yang berupa pencatatan data.
	*Subroutine*	Digunakan untuk menggambarkan hubungan proses dari satu proses ke proses lainnya.
	*Flowline*	Digunakan untuk menggambarkan hubungan proses dari satu proses ke proses lainnya.
	*Page Connector*	Digunakan untuk menghubungkan diagram diagram alur yang terputus dimana bagian tersebut masih berada pada halaman yang sama.
	*Entity*	Kumpulan objek atau sesuatu yang dapat dibedakan atau dapat diidentifikasikan secara unik.

	*Relationship*	Hubungan yang terjadi antara satu entitas atau lebih.
	*Atribute*	Karakteristik dalam entity atau relationship yang mengerjakan penjelasan detail tentang entity atau relationship atau dengan katalain adalah kumpulan elemen data yang membentuk suatu entitas.
	*Atribute Primary Key*	Atribut yang digunakan untuk menentukan suatu *entity* secara unik atau sebagai atribut kunci.

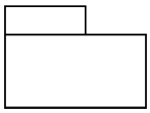
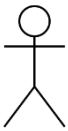
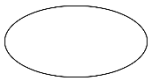
2. Unified Modeling Language (UML)

UML merupakan bahasa pemodelan standar visual yang digunakan untuk menspesifikasikan, memvisualisasikan, membangun, dan mendokumentasikan artefak sistem perangkat lunak berorientasi objek, dengan notasi grafis yang independen dari platform untuk menggambarkan struktur, perilaku, dan arsitektur sistem tanpa mengubah esensi konseptual modelnya, karena UML pada dasarnya adalah serangkaian diagram standar seperti use case, *class*, dan *Sequence* yang bersifat deklaratif (Noneng Marthiawati et al., 2024). Berikut adalah beberapa jenis UML:

a. *Use Case Diagram*

Use Case Diagram adalah jenis diagram UML yang menunjukkan cara kerja suatu sistem dengan menggambarkan interaksi antara berbagai pihak atau entitas dengan fitur-fiturnya. Diagram ini berfungsi sebagai gambaran yang jelas dan tetap tentang apa yang seharusnya dilakukan oleh sistem, dengan menjelaskan cara pengguna akan menggunakannya secara langsung dan jelas (Siska Narulita et al., 2024).

Tabel 2. 2 Use Case Diagram (Sumber : https://app.diagrams.net/)

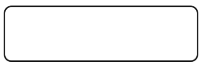
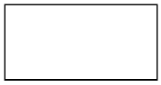
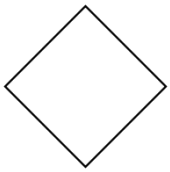
Gambar	Nama	Fungsi
	*Package*	Menambahkan paket baru dalam diagram
	*Actor*	Menambah aktor dalam diagram
	*Use case*	Menambahkan *Use Case* pada diagram
	Unidirectional association	Menggambarkan relasi antara aktor dengan use case
	*Dependencies or Instantiates*	Menggambarkan kebergantungan (dependencies) antar item dalam diagram

b. *Activity Diagram*

Activity Diagram merupakan diagram UML yang berorientasi pada alur kerja, menggunakan simbol diagram alur untuk menggambarkan secara grafis perilaku prosedur bisnis atau fungsi sistem, termasuk aktivitas yang diwakili oleh persegi panjang berlekuk,

keputusan yang ditandai dengan berlian, dan transisi yang ditunjukkan oleh panah (Siska Narulita et al., 2024).

Tabel 2. 3 Activity Diagram (Sumber : https://app.diagrams.net/)

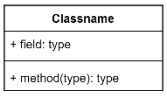
Gambar	*Nama*	*Fungsi*
	State	*Menambahkan state untuk suatu objek*
	Activity	*Menambahkan aktivitas baru pada diagram*
	Start state	*Memperlihatkan di mana aliran kerja berawal*
	End State	*Memperlihatkan di mana aliran kerja berakhir*
	State transition	*Menambah transisi dari suatu aktivitas ke aktivitas yang lainnya*
	Descision points	Menambahkan titik keputusan pada aliran kerja

c. *Class Diagram*

Class Diagram berfungsi sebagai diagram UML struktural, menggambarkan desain tetap dari suatu sistem yang dibangun berdasarkan objek dengan menampilkan kelas, properti mereka, aksi mereka, dan hubungan di antara mereka, seperti asosiasi, agregasi, komposisi, generalisasi, dan ketergantungan; penggambaran ini dapat

dipahami secara konsisten oleh tim pengembangan yang berbeda tanpa mengubah ide inti dari model, karena pada dasarnya berfungsi sebagai rencana untuk basis data relasional atau struktur dasar untuk kode deklaratif (Siska Narulita et al., 2024).

Tabel 2. 4 Class Diagram (Sumber : https://app.diagrams.net/)

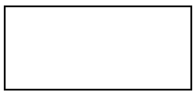
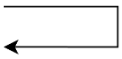
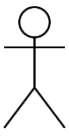
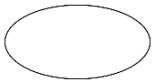
Gambar	Nama	Fungsi
	*Class*	Menambahkan kelas baru pada diagram
	*Interface*	Menambahkan kelas antarmuka (*interface*) pada diagram
	Association	Menggambar relasi asosiasi
	Association Class	Menghubungkan kelas asosiasi (*association class*) pada suatu relasi asosiasi
	Generalization	Menggambarkan relasi generalisasi
	Aggregation	Menggambarkan relasi agregasi

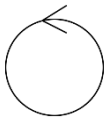
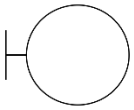
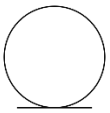
d. *Sequence Diagram*

Sequence Diagram merupakan jenis diagram interaksi UML yang menggambarkan bagaimana objek atau peserta dalam suatu sistem berinteraksi secara berurutan seiring waktu menggunakan garis hidup vertikal, pesan horizontal yang dapat bersifat sinkron atau asinkron, bilah aktivasi, dan fragmen seperti alt, opt, atau *loop*; hal ini memastikan bahwa berbagai tim dapat dengan mudah memahami urutan peristiwa

tanpa mengubah urutan eksekusi dasar, karena diagram ini berfungsi sebagai demonstrasi langsung dari situasi kasus penggunaan yang telah ditentukan sebelumnya (Siska Narulita et al., 2024)

Tabel 2. 5 Sequence Diagram (Sumber : https://app.diagrams.net/)

Gambar	Nama	Fungsi
	*Object*	Menambahkan objek baru pada diagram
	*Object Message*	Menggambar pesan (*message*) antar dua objek
	*Message to self*	Menggambar pesan (*message*) yang menuju dirinya sendiri
	*Return Message*	Menggambarkan pengembalian dari pemanggilan prosedur
	*Destruction Marker*	Memperlihatkan saat objek tertentu dihancurkan
	*Actor*	Menggambarkan aktor pada diagram kelas
	*Use Case*	Menggambarkan *Use Case* pada diagram kelas

	*Control*	Menggambarkan unsur kendali pada diagram
	*Boundary*	Menambahkan kelas batasan (*boundary*) pada diagram
	*Entity*	Menambahkan kelas entitas (*entity*) pada diagram

3. Rich Picture

Rich Picture merupakan salah satu teknik dalam pendekatan *Soft Systems Methodology* (SSM) yang digunakan untuk menggambarkan situasi permasalahan secara menyeluruh dalam bentuk visual yang tidak terstruktur. *Rich Picture* digunakan untuk merepresentasikan berbagai elemen dalam suatu sistem, seperti aktor yang terlibat, proses yang terjadi, hubungan antar komponen, serta permasalahan yang muncul, sehingga memberikan pemahaman yang lebih komprehensif terhadap kondisi nyata di lapangan (Septiana & Maulany, 2021)

Berdasarkan penjelasan tersebut, *Rich Picture* merupakan alat visual yang digunakan untuk menggambarkan situasi masalah secara menyeluruh dan kompleks dengan pendekatan yang fleksibel dan tidak terstruktur. Teknik ini bertujuan untuk membantu memahami kondisi sistem, mengidentifikasi permasalahan, serta mendukung proses perencanaan solusi, sehingga pengembangan sistem dapat dilakukan secara lebih tepat dan sesuai dengan kebutuhan yang ada.

2.1.15 *Tools* Perangkat Lunak

1. Google Colaboratory (Google Colab)

Google Colaboratory (Google Colab) merupakan platform berbasis cloud computing yang disediakan oleh Google untuk mendukung proses pengembangan dan eksekusi kode pemrograman, khususnya dalam bidang analisis data, machine learning, dan deep learning. Google Colab memungkinkan pengguna untuk menulis dan menjalankan kode Python secara langsung melalui browser tanpa perlu melakukan instalasi perangkat lunak tambahan. Platform ini mengubah kebutuhan komputasi menjadi lebih praktis dengan menyediakan lingkungan kerja yang siap pakai, lengkap dengan dukungan pustaka populer seperti TensorFlow, PyTorch, NumPy, dan Pandas, sehingga mempermudah proses pengembangan dan pengujian model (Octarina et al., 2025)

Berdasarkan penjelasan tersebut, Google Colaboratory (Google Colab) merupakan suatu platform yang dirancang untuk mempermudah proses pengembangan, pengujian, dan implementasi kode pemrograman secara berbasis *cloud* dengan lingkungan yang fleksibel dan terintegrasi. Platform ini tidak hanya digunakan untuk membangun proyek baru, tetapi juga untuk mengoptimalkan proses komputasi dan kolaborasi dalam pengembangan sistem, sehingga dapat berjalan secara efektif sesuai dengan kebutuhan pengguna dan perkembangan teknologi saat ini.

2. Google Drive

Google Drive merupakan layanan penyimpanan berbasis cloud yang dikembangkan oleh Google untuk mendukung pengelolaan, penyimpanan, dan berbagi data secara online. Google Drive memungkinkan pengguna untuk menyimpan berbagai jenis file seperti dokumen, gambar, video, dan data lainnya dalam satu platform yang dapat diakses kapan saja dan di mana saja melalui koneksi internet. Layanan ini mengubah cara penyimpanan tradisional menjadi lebih fleksibel dan terintegrasi, serta menyediakan fitur sinkronisasi otomatis

yang memudahkan pengguna dalam mengelola data secara efisien (Salsabila et al., 2023)

Berdasarkan penjelasan tersebut, Google Drive merupakan suatu layanan penyimpanan berbasis *cloud* yang dirancang untuk mempermudah proses penyimpanan, pengelolaan, dan distribusi data secara digital. Layanan ini tidak hanya digunakan untuk menyimpan data baru, tetapi juga untuk meningkatkan efisiensi pengelolaan data yang sudah ada melalui sistem yang terintegrasi, sehingga dapat diakses dan digunakan secara optimal sesuai dengan kebutuhan pengguna.

3. Visual Studio Code

Visual Studio Code merupakan code editor yang dikembangkan oleh Microsoft untuk mendukung proses pengembangan perangkat lunak secara efisien dan fleksibel. Visual Studio Code dirancang sebagai lingkungan pengembangan yang ringan namun мощ—lengkap dengan berbagai fitur seperti syntax highlighting, *debugging*, code completion (IntelliSense), serta integrasi dengan berbagai bahasa pemrograman seperti Python, Java, JavaScript, dan lainnya. Perangkat lunak ini mengubah kebutuhan pengkodean menjadi lebih terstruktur dan mudah dikelola melalui antarmuka yang intuitif serta dukungan ekstensi yang luas, sehingga mempermudah pengembang dalam membangun dan mengelola aplikasi (Nur Aeni Hidayah & Rofiqoh, 2024).

Berdasarkan penjelasan tersebut, Visual Studio Code merupakan suatu alat pengembangan perangkat lunak yang dirancang untuk mempermudah proses penulisan, pengujian, dan pengelolaan kode secara efisien dan terintegrasi. Aplikasi ini tidak hanya digunakan untuk membangun program baru, tetapi juga untuk meningkatkan produktivitas dan kualitas pengembangan sistem yang sudah ada, sehingga dapat

berjalan secara optimal sesuai dengan kebutuhan pengguna dan perkembangan teknologi saat ini.

4. DrawIO

Draw.io merupakan alat berbasis web yang digunakan untuk membuat berbagai jenis diagram secara visual guna mendukung proses perancangan sistem. Draw.io memungkinkan pengguna untuk merancang diagram seperti flowchart, UML, ERD, dan diagram arsitektur sistem dengan mudah melalui antarmuka grafis yang intuitif. Alat ini mengubah proses perancangan yang kompleks menjadi lebih sederhana dengan menyediakan berbagai elemen visual seperti simbol, garis, dan bentuk yang dapat disusun secara terstruktur, sehingga mempermudah dalam menggambarkan alur dan struktur sistem secara jelas (Noneng Marthiawati et al., 2024)

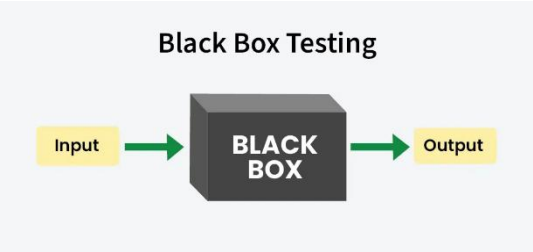
Berdasarkan penjelasan tersebut, Draw.io merupakan suatu alat bantu visual yang dirancang untuk mempermudah proses perancangan dan dokumentasi sistem melalui pembuatan diagram yang terstruktur dan informatif. Alat ini tidak hanya digunakan untuk membuat diagram baru, tetapi juga untuk menyempurnakan representasi sistem yang sudah ada, sehingga dapat mendukung proses pengembangan sistem secara lebih efektif dan sesuai dengan kebutuhan pengguna.

2.1.16 Pengujian Perangkat Lunak

1. Black Box Testing

Black Box Testing merupakan metode pengujian perangkat lunak yang berfokus pada validasi fungsionalitas sistem melalui pemeriksaan *input* dan *output* yang diharapkan tanpa mengakses atau menganalisis struktur kode internal, struktur data, atau algoritma implementasi, karena

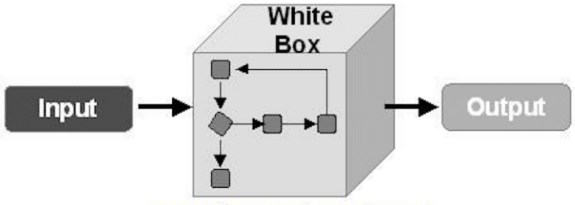
pendekatan ini memperlakukan modul sebagai kotak hitam yang opaque sepenuhnya dari perspektif penguji (Praniffa et al., 2023).



Gambar 2. 16 *Black Box Testing*

2. White Box Testing

White Box Testing merupakan metode pengujian perangkat lunak yang berfokus pada analisis struktur internal kode sumber, jalur eksekusi, dan logika program untuk memverifikasi cakupan statement, branch, serta kondisi tanpa mengubah esensi kode asli, karena pendekatan ini memperlakukan sistem sebagai kotak transparan yang memungkinkan penguji menelusuri alur data dan kontrol flow secara eksplisit (Praniffa et al., 2023)



Gambar 2. 17 White Box

2.1.17 Evaluasi Model

Evaluasi model merupakan tahap penting dalam proses pengembangan sistem klasifikasi yang bertujuan untuk mengukur sejauh mana model mampu memberikan prediksi yang akurat dan sesuai dengan data sebenarnya. Salah satu metode yang umum digunakan adalah *Confusion Matrix*, yaitu tabel yang digunakan untuk membandingkan hasil prediksi model dengan nilai aktual. Melalui metode ini, kinerja model dapat dianalisis secara lebih rinci berdasarkan jumlah prediksi yang benar maupun salah dalam setiap kategori. *Confusion Matrix* terdiri dari empat komponen utama, yaitu True Positive (TP),

True Negative (TN), False Positive (FP), dan False Negative (FN). TP menunjukkan jumlah data positif yang diprediksi dengan benar, TN menunjukkan data negatif yang diprediksi dengan benar, FP adalah data negatif yang salah diprediksi sebagai positif, sedangkan FN adalah data positif yang salah diprediksi sebagai negative (Hidayah & Dodiman, 2024). Berikut merupakan table dari *Confusion Matrix*:

Tabel 2. 6 *Confusion Matrix*

Actual	**Predicted**	
	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Negative (FP)	True Negative (TN)

Berdasarkan nilai-nilai tersebut, beberapa metrik evaluasi dapat dihitung untuk menilai performa model, yaitu sebagai berikut:

1. *Accuracy*

Accuracy merupakan ukuran tingkat keakuratan model dalam mengklasifikasikan data secara keseluruhan.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

2. *Precision*

Precision mengukur tingkat ketepatan prediksi positif yang dihasilkan oleh model.

$$Precision = \frac{TP}{TP + FP}$$

3. *Recall*

Recall (Sensitivity) mengukur kemampuan model dalam mendeteksi seluruh data positif yang sebenarnya.

$$Recall = \frac{TP}{TP + FN}$$

4. *F1-Score*

F1-Score merupakan rata-rata harmonis antara *precision* dan *recall* yang digunakan untuk menyeimbangkan kedua metrik tersebut.

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

2.1.18 User Acceptance *Test*

User acceptance test (UAT) merupakan salah satu metode pengujian yang digunakan untuk mengetahui tingkat penerimaan pengguna terhadap sistem atau aplikasi yang telah dikembangkan. Pengujian ini dilakukan dengan melibatkan pengguna secara langsung untuk mencoba aplikasi, menjalankan beberapa tugas, kemudian memberikan penilaian melalui kuesioner (Alfian et al., 2022). Dalam penelitian (Agusti & Alfian, 2023), pengujian sistem dilakukan menggunakan *black box testing* dan *user acceptance test*. Hasil pengujian *black box* menunjukkan nilai 100% sesuai pada seluruh tahap pengujian, sedangkan hasil UAT yang dilakukan kepada 20 responden memperoleh nilai 85,4%, sehingga aplikasi dinyatakan sangat layak digunakan. Hal ini menunjukkan bahwa UAT berperan penting dalam menilai kelayakan aplikasi berdasarkan pengalaman dan penilaian langsung dari pengguna akhir.

Berdasarkan penjelasan tersebut, *User acceptance test* merupakan metode pengujian yang bertujuan untuk mengetahui sejauh mana aplikasi dapat diterima oleh pengguna akhir berdasarkan aspek tampilan, kemudahan penggunaan, fungsi, dan efisiensi. Dalam penelitian ini, UAT digunakan untuk mengevaluasi aplikasi klasifikasi penyakit pada daun tomat berbasis Android dengan melibatkan pengguna secara langsung dalam mencoba fitur aplikasi, seperti pengambilan gambar, proses klasifikasi, dan tampilan hasil deteksi. Hasil dari pengujian ini menjadi dasar untuk menentukan apakah aplikasi telah sesuai dengan kebutuhan pengguna, mudah digunakan, serta layak diterapkan sebagai alat bantu dalam mengklasifikasikan penyakit pada daun tomat.

2.2 Penelitian Sebelumnya *(Previous Work)*

Penelitian-penelitian terdahulu yang relevan dengan penelitian yang dilakukan oleh penyusun saat ini. Terhadap beberapa penelitian tersebut diharapkan dapat menemukan alternatif solusi dengan menggunakan, memodifikasi dan atau menemukan model baru.

Penelitian sebelumnya ini menjadi pembanding atau perbedaan antara penelitian sebelumnya yang pernah ada dan yang akan peneliti lakukan sebagai solusi untuk melakukan penelitian.

Tabel 2. 7 Perbandingan Penelitian Sebelumnya

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/penelitian	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
1.	*Mobile Application for Tomato Plant Leaf Disease Detection Using a Dense Convoluti	Penyakit daun tanaman tomat sulit diidentifikasi secara visual oleh petani. Kesamaan	Mengklasifikasikan penyakit daun tomat menggunakan arsitektur DenseNet dan mengimplementasikan model terbaik ke dalam aplikasi mobile	1) *Dataset*: 10.000 gambar (1.000/kelas, 10 kelas penyakit) dari Kaggle. 2) *Preprocessing*: *rescale* (0-1), augmentasi (rotasi 40°, shift, *zoom*).	10.000 gambar daun tomat (*Training*) + 1.000 (*testing*); 10 kelas penyakit.	Akurasi: 95.7%, *F1-Score*: 95.4%. Model terbaik: 2 hidden *layer*, *dropout* 0.4, *train*able dense block 5. Akurasi tertinggi per	*Dataset* terbatas dari kondisi laboratorium (Kaggle). Belum diuji pada gambar	Persamaan: Sama-sama menggunakan CNN dan *dataset* penyakit daun tomat. Menggunakan metrik presisi,

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/peneliti an	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
	onal Network Architectu re (Yulita et al., 2023)	gejala antar penyakit menyebab kan kesalahan diagnosa yang berpotensi merugikan secara finansial.	Android berbahasa Indonesia.	3) Arsitektur: DenseNet dengan 5 dense block; *train*able *layer* pada dense block 5. 4) Hyperparameter : 2 hidden *layer*, *dropout* 0.4, Adam optimizer, lr 0.01. 5) Evaluasi: 10-fold cross-validation, *Confusion matrix*, akurasi, presisi, *recall*, *F1-Score*. 6) Implementasi: Aplikasi Android (Flutter).		kelas: Mosaic Virus (99.49%), terendah: Late Blight (89.90%).	lapangan nyata. Performa model Late Blight rendah karena tingginya false positive.	*recall*, *F1-Score*. Mengembangkan aplikasi mobile Android. Perbedaan: Penelitian terdahulu menggunakan DenseNet; penelitian saat ini menggunakan MobileNetV2. *Dataset* penelitian terdahulu dari Kaggle; penelitian saat ini menggunaka

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/penelitian	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
								n *dataset* PlantVillage yang dikumpulkan sendiri.
2	*Tomato Leaf Disease Classification by Exploiting Transfer learning and Feature Concatenation* (Al-gaashani et al., 2022)	Klasifikasi penyakit daun tomat menggunakan *deep learning* memerlukan data besar dan komputasi tinggi. *Dataset* terbatas dapat menurunkan performa	Mengusulkan metode klasifikasi penyakit daun tomat dengan menggabungkan fitur dari dua model pre-*train*ed (MobileNetV2 dan NASNetMobile) melalui *feature* concatenation dan *machine learning* classifier.	1) *Dataset*: 1.152 gambar PlantVillage (6 kelas: 1 sehat, 5 penyakit; 192 gambar/kelas); *resize* 224×224. 2) *Transfer learning*: MobileNetV2 (V_Mobnet: 62.720 fitur) dan NASNetMobile (V_Nasnet: 51.744 fitur) sebagai fixed	1.152 gambar daun tomat (PlantVillage); 864 *Training*, 288 *testing* (6 kelas).	Proposal 2 (concatenation + MLR + *Kernel* PCA): Akurasi 97%. MLR unggul dibanding SVM dan RF. MobileNetV2 lebih baik dari NASNetMobile dalam ekstraksi fitur. Waktu *Training* dan *testing* lebih rendah dibanding	*Dataset* sangat terbatas (1.152 gambar). Gambar dari kondisi laboratorium, belum diuji di lapangan. Tipe penyakit terbatas pada 5 jenis saja.	Persamaan: Sama-sama menggunakan *transfer learning* berbasis CNN, *dataset* PlantVillage, dan metrik presisi/*recall*/F1. Perbedaan: Penelitian terdahulu menggunakan feature concatenation

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/peneliti an	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
		model deep learning secara drastis.		*feature extractor*. 3) Reduksi dimensi: *Kernel* PCA (linear, RBF, polynomial) 4) Proposal 2: Concatenation fitur V_Mobnet + V_Nasnet. 5) Classifier: SVM, RF, MLR (grid search hyperparameter). 6) Evaluasi: Akurasi, presisi, *recall*, *F1-Score*, AUC-ROC.		metode deep learning penuh.		MobileNetV2+NASNetMobile lalu classifier ML (MLR); penelitian saat ini menggunakan fine-tuning end-to-end MobileNetV2. *Dataset* penelitian terdahulu hanya 1.152 gambar (6 kelas); penelitian saat ini lebih banyak kelas penyakit tomat.

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/penelitian	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
3	*A ResNet50-DPA Model for Tomato Leaf Disease Identification* (Liang & Jiang, 2023).	Identifikasi penyakit daun tomat sulit karena perbedaan visual antar jenis penyakit sangat kecil. Metode CNN konvensional cenderung kehilangan fitur penting dan menghasilkan akurasi	Mengusulkan model ResNet50-DPA (Dual-Path Attention) untuk identifikasi penyakit daun tomat dengan meningkatkan kemampuan ekstraksi fitur melalui atrous *convolution* dan mekanisme attention.	1) *Dataset*: PlantVillage, 22.930 gambar (10 kelas; 18.345 *Training*, 4.585 *testing*); augmentasi imgaug (emboss, equalization, rotasi, clipping, Gaussian noise, *flip*). 2) Model: ResNet50 + Cascaded Atrous *Convolution* (rate=1 & 2) menggantikan conv 7×7	22.930 gambar daun tomat PlantVillage; 10 kelas penyakit.	ResNet50-DPA: Akurasi 99.28%, Presisi 99.29%, *Recall* 99.28%, F1 99.28%. Unggul dibanding AlexNet (74.15%), MobileNetV2 (95.00%), DenseNet121 (97.97%), ResNet34 (96.97%), ResNet50 (97.43%), ResNet101 (97.60%). Leaf Mold & Mosaic Virus:	Skenario aplikasi masih terbatas. Belum diuji pada gambar lapangan nyata. Generalisasi dan kecepatan komputasi perlu ditingkatkan. *Leaf Spot* diidentifikasi sebagai spider mites (akurasi	Persamaan: Sama-sama menggunakan CNN dengan *transfer learning* berbasis ResNet untuk klasifikasi penyakit daun tomat. Menggunakan *dataset* PlantVillage dan metrik akurasi/presisi/*recall*/F1. Perbedaan: Penelitian terdahulu menggunakan

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/peneliti an	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
		rendah.		pertama. 3) DPA mechanism: Channel Attention (MaxPool+StoPool+AvgPool+Conv1D) + Spatial Attention (MaxPool+StoPool+AvgPool+Conv7×7). 4) Soft *pooling* menggantikan max *pooling layer*. 5) Optimizer: Adam (lr=0.001), *batch size* 32, 200 *epoch*. 6) Evaluasi:		akurasi 100%.	hanya 97.09%).	ResNet50+DPA (attention mechanism baru); penelitian saat ini menggunakan MobileNetV2 yang lebih ringan untuk implementasi mobile. Penelitian terdahulu tidak mengembangkan aplikasi mobile; penelitian saat ini mengembangkan aplikasi

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/peneliti an	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
				Akurasi, presisi, *recall*, *F1-Score*, *Confusion matrix*, Grad-CAM.				Android.
4	Klasifikasi Tingkat Kematangan Buah Nanas Menggunakan Metode *Deep Learning* (Liandaputra & Zahra, 2024).	Penentuan tingkat kematangan buah nanas masih dilakukan secara manual sehingga tidak konsisten dan berpotensi merugikan finansial petani	Mengklasifikasikan tingkat kematangan buah nanas (mentah, kurang matang, matang, terlalu matang) menggunakan metode *deep learning* berbasis CNN.	1) Pengumpulan *dataset* 400 gambar nanas varietas Tuatunu dari perkebunan rakyat. 2) *Preprocessing*: anotasi VGGAnnotator, cropping, *resize* 224×224. 3) Augmentasi: geometrik (rotasi 15°, 30°, translasi)	400 gambar nanas (100/kelas) dari 4 lokasi di Desa Tua Tunu, Pangkalpinang, Bangka Belitung.	MobileNetV2 (GAP+Dense+ *Dropout*) + augmentasi kombinasi terbaik. Akurasi *Training*: 99,07%, validasi: 93%, uji: 90,49%, presisi: 0.91, *recall*: 0.91, F1: 0.91.	*Dataset* terbatas (400 gambar) dari satu varietas nanas. Belum diimplementasikan pada perangkat mobile/embedded system. Kondisi cahaya	Persamaan: Sama-sama menggunakan CNN untuk klasifikasi penyakit tanaman dari citra. Objek penelitian sama-sama tanaman hortikultura. Menggunakan metrik akurasi untuk evaluasi. Perbedaan:

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/peneliti an	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
				dan fotometrik (brightness, Gaussian blur). 4) Perbandingan arsitektur: AlexNet, VGG16, MobileNetV1, MobileNetV2. 5) Evaluasi: *Confusion matrix*, presisi, *recall*, *F1-Score*.			dan sudut terbatas.	Penelitian terdahulu mengklasifikasi penyakit buah cabai merah; penelitian saat ini mengklasifikasi penyakit daun tomat. Penelitian terdahulu menggunakan arsitektur CNN kustom; penelitian saat ini menggunakan MobileNetV2.

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/penelitian	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
5	Perbandingan Metode *EfficientNet*-B3 dan MobileNet-V2 Untuk Identifikasi Jenis Buah-buahan Menggunakan Fitur Daun (Zaelani & Miftahuddin, 2022).	Masyarakat umum, terutama anak-anak, kesulitan membedakan jenis buah-buahan hanya dari bentuk daunnya. Belum ada sistem otomatis yang efektif untuk identifikasi jenis buah dari citra	Membandingkan performa arsitektur EfficientNet-B3 dan MobileNetV2 dalam mengidentifikasi 7 jenis buah-buahan berdasarkan citra daun untuk mendapatkan akurasi terbaik.	1) *Dataset* 4.312 gambar daun dari 7 kelas (jagung, mangga, tomat, mulberry, pisang, jeruk, jambu) dari Kaggle. 2) *Preprocessing*: *rescale*, rotasi, *flip*. 3) Augmentasi data untuk mengurangi *overfitting*. 4) *Training* & validasi dengan rasio 80:10:10. 5) Perbandingan EfficientNet-B3 vs	4.312 gambar daun dari 7 kelas. *Training* EfficientNetB3: 3.232, *testing*: 844. *Training* MobileNetV2: 3.833, *testing*: 698.	MobileNetV2 *epoch* 20, optimizer Adam unggul dengan akurasi 99% dan *loss* 0.1046. EfficientNet-B3 terbaik hanya 30% (*epoch* 50, Adamax).	EfficientNet-B3 menunjukkan performa sangat rendah (23-30%) untuk *dataset* daun ini. *Dataset* hanya mencakup 7 jenis buah dari Kaggle, tidak merepresentasikan kondisi lapangan nyata.	Persamaan: Sama-sama menggunakan CNN dengan pendekatan *transfer learning* untuk klasifikasi citra tanaman. Objek penelitian sama-sama tomat. Menggunakan metrik presisi, *recall*, *F1-Score*. Perbedaan: Penelitian

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/peneliti an	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
		daun.		MobileNetV2 dengan variasi *epoch* (20/50) dan optimizer (Adam, Adamax, SGD). 6) Evaluasi: *Precision*, *Recall*, F-Score, *Confusion Matrix*.				terdahulu mengklasifikasi kematangan buah tomat; penelitian saat ini mengklasifikasi penyakit daun tomat. Penelitian terdahulu menggunakan EfficientNet-B7; penelitian saat ini menggunakan MobileNetV2. *Dataset* penelitian

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/penelitian	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
								terdahulu dari Kaggle; penelitian saat ini menggunakan *dataset* sendiri.
6	Klasifikasi Penyakit Antraknosa Pada Cabai Merah Teropong "*Inko Hot*" Dengan Metode *Convolutional Neural Network* (Avianto	Petani cabai merah sering mengalami kerugian akibat penyakit antraknosa yang dipicu jamur Colletotrichum sp., berpotensi mengakib	Mengembangkan sistem klasifikasi berbasis CNN untuk mengidentifikasi penyakit antraknosa pada buah cabai merah varietas "Inko Hot" secara otomatis dan efisien.	1) Pengumpulan 1.000 data citra cabai merah (500 normal, 500 antraknosa) dari kebun di Desa Widoro, Sleman, Yogyakarta; ditambah 500 citra sayuran lain untuk penanganan *input* non-cabai.	1.500 data total: 500 citra cabai normal, 500 cabai antraknosa, 500 citra sayuran lain (kacang panjang, buncis, tomat, wortel).	Model terbaik dengan optimizer Nadam, 50 *epoch*: akurasi 97%, *loss* 6,45%. Ketepatan prediksi pada data baru: 83,33% (5/6 data benar).	*Dataset* terbatas (1.000 citra cabai) dari satu kebun. Hanya mengklasifikasi 2 kelas (normal vs antraknosa). Belum	Persamaan: Sama-sama menggunakan CNN dengan *transfer learning* untuk identifikasi/klasifikasi penyakit daun tanaman. Sama-sama menggunakan

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/peneliti an	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
	& Handayani, 2023).	atkan kegagalan panen hingga 80%, namun klasifikasi penyakit masih manual.		2) *Preprocessing*: *resize* 150×150, augmentasi (rotasi 0-30°, *shear* 0.3, *flip* horizontal/vertikal, *zoom* 0.1, shift 0.2). 3) Arsitektur CNN kustom (Sequential): Conv2D×3, Max*Pooling*×3, *Flatten*, Dense, *Dropout*. 4) *Training*: pembagian 80% latih/20% validasi, berbagai optimizer			diuji pada varietas cabai lain atau kondisi lapangan yang berbeda.	MobileNetV2. Menggunakan metrik presisi, *recall*, *F1-Score*, dan akurasi sebagai evaluasi. Perbedaan: Penelitian terdahulu (internasional) mengklasifikasi 38 kelas penyakit dari 14 spesies; penelitian saat ini fokus pada penyakit

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/peneliti an	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
				(Adadelta, Adagrad, Adam, Adamax, Nadam, RMSprop, SGD), 50 *epoch*. 5) Evaluasi: akurasi, *loss*, *testing* pada 6 citra baru.				daun tomat saja. *Dataset* penelitian terdahulu dari PlantVillage (laboratorium); penelitian saat ini menggunakan *dataset* yang dikumpulkan sendiri. Penelitian saat ini mengembangkan aplikasi Android untuk implementasi di lapangan.

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/penelitian	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
7	Klasifikasi Penyakit Daun Pisang Menggunakan *Convolutional Neural Network* (CNN) (Pratama et al., 2024).	Petani kesulitan mengidentifikasi jenis penyakit daun pisang (cordana, pestalotiopsis, sigatoka) secara akurat karena kemiripan visual gejala dan kurangnya pengetahuan tentang patogen.	Membangun model klasifikasi penyakit daun pisang menggunakan CNN dengan arsitektur VGG19 dan *transfer learning* untuk membantu petani mengidentifikasi penyakit secara otomatis.	1) *Dataset* 1.600 gambar dari Mendeley Data (BSMRAU), 4 kelas (cordana, healthy, pestalotiopsis, sigatoka), 400 gambar/kelas. 2) *Preprocessing*: crop, *resize* 224×224, augmentasi data. 3) Arsitektur: VGG19 dengan *transfer learning* (Google Colab). 4) Pembagian data: 70:20:10	1.600 gambar daun pisang dari Mendeley Data (BSMRAU), 4 kelas, 400 gambar/kelas. Data uji: 80 gambar (20/kelas).	Model terbaik (rasio 70:20:10, *epoch* 80): akurasi *Training* 92%, validasi 94%. Hasil uji: akurasi 82%, presisi 85%, *recall* 83%, *F1-Score* 82%.	*Dataset* terbatas (1.600 gambar) dari satu sumber. Akurasi uji (82%) lebih rendah dari akurasi validasi (94%) yang menunjukkan adanya gap generalisasi. Belum diuji pada kondisi lapangan	

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/penelitian	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
				dan 80:10:10. 5) Regularisasi: *dropout* untuk mengatasi *overfitting*. 6) Evaluasi: *Confusion matrix*, akurasi, presisi, *recall*, *F1-Score*. Uji pada 80 gambar *test*.			nyata di kebun.	
8	*Identification of Plant-Leaf Diseases Using CNN and Transfer-Learning Approach*	Identifikasi penyakit tanaman secara manual di daerah pedesaan membutuhkan	Mengimplementasikan model CNN berbasis *transfer learning* (InceptionV3, InceptionResNetV2, MobileNetV2, EfficientNetB0) untuk	1) *Dataset*: PlantVillage (54.305 gambar, 38 kelas penyakit, 14 spesies tanaman) dalam format warna, grayscale, dan	54.305 gambar daun dari PlantVillage *dataset*, mencakup 38 kelas (14 spesies tanaman), dengan 3	EfficientNetB0 terbaik: akurasi 99,56%, presisi 0.9953, *recall* 0.9971, *F1-Score* 0.9961, waktu 545 detik/*epoch*.	*Dataset* hanya dari kondisi laboratorium (PlantVillage), bukan kondisi	

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/penelitian	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
	(Hassan et al., 2021).	keahlian khusus dan memakan waktu. Model CNN standar memerlukan parameter yang banyak dan biaya komputasi tinggi, sehingga kurang efisien untuk implementasi praktis.	mengidentifikasi penyakit tanaman dari citra daun dengan akurasi tinggi dan waktu pelatihan lebih singkat.	tersegmentasi. 2) Arsitektur: InceptionV3 dan InceptionResNetV2 dimodifikasi dengan *depthwise separable convolution*; MobileNetV2 dan EfficientNetB0 menggunakan *transfer learning* dari ImageNet. 3) Penambahan lapisan: activation layer, batch-normalization,	format citra: warna, grayscale, dan tersegmentasi.	MobileNetV2: akurasi 97,02%, presisi 0.9732, *recall* 0.9962, F1 0.9845, waktu 565 detik/*epoch*. *Depthwise separable convolution* berhasil mengurangi jumlah parameter secara signifikan.	lapangan nyata. Saat diuji dengan gambar nyata di lapangan, akurasi turun drastis (sekitar 31-33%). Penelitian belum diimplementasikan pada perangkat mobile/embedded secara langsung.	

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/penelitian	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
				dropout (0.2-0.8), dan regularisasi L1/L2 pada dense layer. 4) Pembagian data: 80:20, 70:30, dan 60:40 untuk evaluasi ketahanan model. 5) Parameter *Training*: *batch size* 32-180, *learning rate* 0.01-0.0001, *epoch* 30-50. 6) Evaluasi: akurasi, *loss*, presisi, *recall*, *F1-Score*, dan				

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/penelitian	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
				waktu *Training* per *epoch*.				
9	*A VGG-19 Model with Transfer learning and Image Segmentation for Classification of Tomato Leaf Disease* (T.-H. Nguyen et al., 2022).	Penyakit daun tomat dapat menurunkan kualitas dan kuantitas panen. Deteksi penyakit yang masih manual kurang efisien. Model deep learning tanpa	Mengklasifikasikan 10 jenis penyakit daun tomat (9 penyakit + 1 sehat) menggunakan VGG-19 dengan *transfer learning*, didahului segmentasi citra berbasis ruang warna HSV untuk meningkatkan akurasi dan mempercepat waktu pelatihan.	1) *Dataset*: PlantVillage (16.010 gambar daun tomat, 10 kelas: 9 penyakit dan 1 sehat), *resize* ke 224×224. 2) Segmentasi citra: konversi *RGB* ke *HSV*, threshold histogram (H: 0.18-0.5, S: 0.1-1.0, V: 0.0-0.8), morfologi untuk mengekstrak area daun dengan latar	16.010 gambar daun tomat dari PlantVillage, 10 kelas. Pembagian: 80% *Training*+validasi (64% *train*, 16% validasi) dan 20% *testing*.	VGG-19 + segmentasi HSV terbaik: ACC 99,72%, SEN 99,69%, SPE 99,90%, PPV 99,49%, F1S 99,59%. Tanpa segmentasi: ACC 99,63%. Segmentasi mempercepat *Training* ($2{,}75\times10^5$ vs $2{,}98\times10^5$ detik) dan *testing* (29,30 vs 35,27 detik). VGG-19	*Dataset* hanya dari PlantVillage (kondisi laboratorium), belum diuji pada gambar lapangan nyata secara sistematis. Jumlah gambar tiap kelas tidak seimbang	Persamaan: Sama-sama menggunakan CNN dengan *transfer learning* untuk klasifikasi penyakit daun tomat. Sama-sama menggunakan *dataset* PlantVillage. Menggunakan metrik *Confusion matrix*, presisi/PPV, *recall*/SEN,

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/peneliti an	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
		segmentasi membutuhkan waktu *Training* lebih lama dan latar belakang gambar yang bervariasi mempengaruhi akurasi klasifikasi.		hitam seragam. 3) Model: VGG-19 (16 conv layer + 3 *Fully connected layer*) dengan *transfer learning*; conv *layer* dibekukan (freeze), *Fully connected layer* dilatih ulang. 4) Parameter: *epoch* 300, *learning rate* 0.00001, *batch size* 60, fungsi aktivasi *Softmax*. 5) Perbandingan: VGG-19 vs		mengungguli AlexNet (99,16%), ResNet50 (99,19%), dan GoogleNet (99,38%).	(mis. Mosaic virus hanya 373 gambar) yang berpotensi menurunkan SEN kelas minoritas. Segmentasi HSV bergantung pada nilai threshold yang ditentukan secara manual, sehingga	dan *F1-Score*. Perbedaan: Penelitian terdahulu (internasional) menggunakan VGG-19 dengan segmentasi HSV; penelitian saat ini menggunakan MobileNetV2 tanpa segmentasi. Penelitian terdahulu mengklasifikasi 10 kelas

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/penelitian	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
				AlexNet, GoogleNet, ResNet50. 6) Evaluasi: *Confusion matrix*, ACC, SEN, SPE, PPV, *F1-Score*; perbandingan waktu *Training*/*testing* antara citra tersegmentasi dan tidak.			kurang robust untuk kondisi cahaya berbeda.	penyakit tomat; penelitian saat ini mengklasifikasi 3 jenis penyakit. Penelitian saat ini mengembangkan aplikasi Android untuk digunakan petani di lapangan.
10	*Development of a Convolutional Neural Network Method*	Klasifikasi tingkat kematangan buah tomat secara manual	Mengembangkan metode CNN yang dimodifikasi (pengurangan layer, penambahan batch	1) *Dataset*: 500 gambar tomat varietas Servo (100/kelas, 5 kelas: Unriped, Half-Riped, Riped,	500 gambar asli (100/kelas); setelah augmentasi : 1.739	Model 3 (batch size 16) adalah model terbaik: Akurasi 98%, Presisi 98%, *Recall* 98%,	*Dataset* sangat terbatas (500 gambar, 1 varietas). Gambar	Persamaan: Sama-sama menggunakan metode CNN untuk klasifikasi citra tomat.

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/penelitian	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
	for Classifying Ripeness Levels of Servo Variety Tomatoes (Rosalina et al., 2025).	rentan terhadap human error, tidak efisien, dan membutuhkan banyak tenaga serta waktu. Peningkatan produksi tomat memperburuk kondisi ini dan berpotensi menggang	normalization, dan dropout) untuk meningkatkan akurasi klasifikasi tingkat kematangan buah tomat varietas Servo ke dalam 5 kelas.	Half-Rotten, Rotten) diambil sendiri menggunakan Samsung Galaxy A54, latar putih, jarak 30 cm. 2) *Preprocessing*: *rescale* (0-1), *resize* dari 3056x3056 ke 224x224 piksel. 3) *Split* data: 70% *Training* (350), 20% validasi (100), 10% *testing* (50). 4) Augmentasi: rotasi 40°, horizontal *flip*,	gambar *Training*. *Testing*: 50 gambar.	*F1-Score* 98%, Parameter hanya 2,4 juta (94,15% lebih kecil dari penelitian sebelumnya dengan 42,5 juta parameter). Model 3 lebih stabil dibanding Model 1 & 2 berdasarkan grafik *loss*/*accuracy*. Model 1 mengalami *overfitting* (*train* acc 100% vs val acc 95%).	diambil di kondisi terkontrol (studio box), belum diuji di kondisi lapangan nyata. Kondisi pencahayaan seragam; variasi pencahayaan belum dieksplorasi. Belum diimplementasikan	Menggunakan metrik presisi, *recall*, *F1-Score*, dan akurasi. Menggunakan teknik augmentasi data. Menggunakan optimizer Adam dan *preprocessing rescale* + resize 224x224. Perbedaan: Penelitian terdahulu mengklasifikasikan tingkat

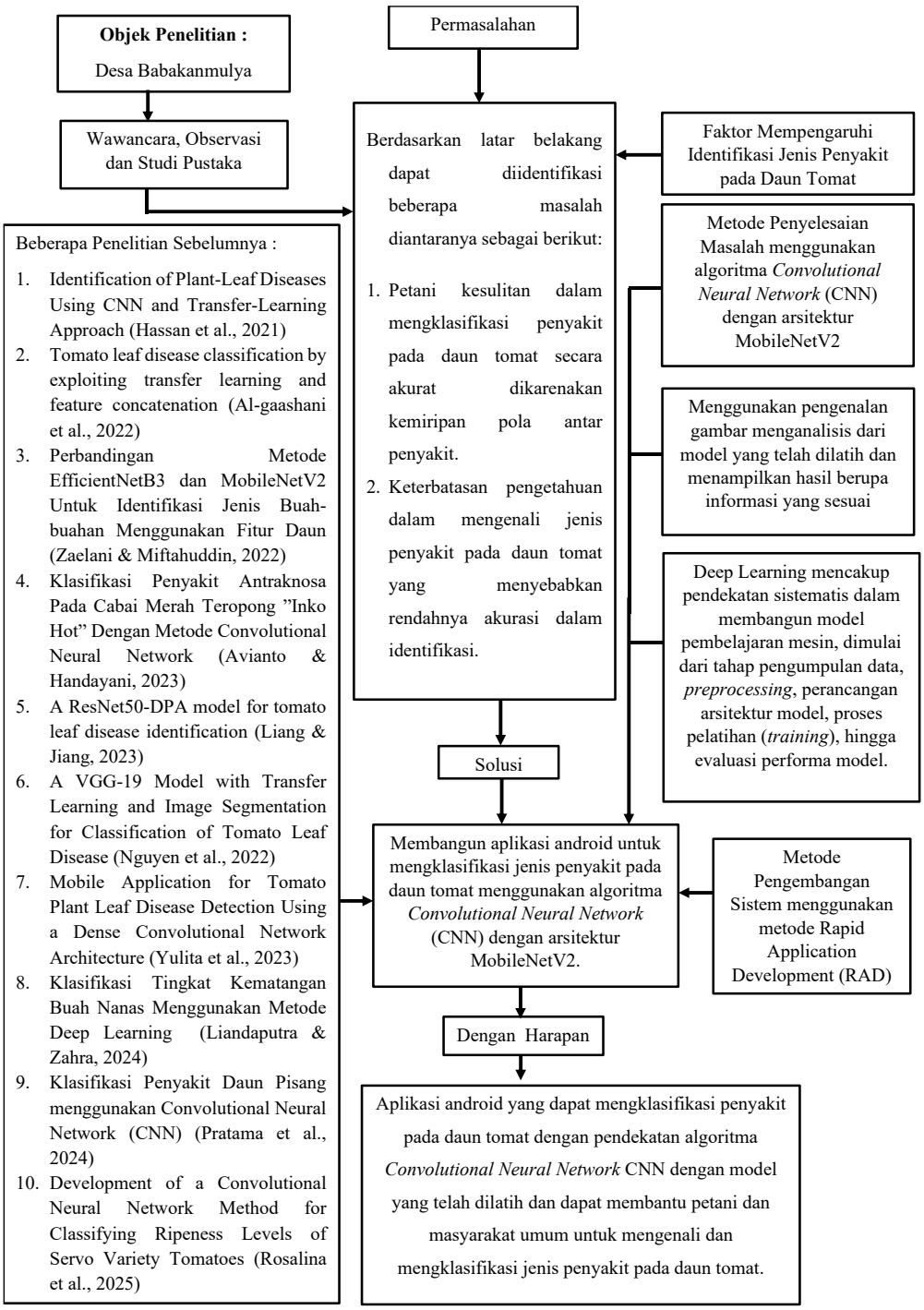
No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/penelitian	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
		gu rantai pasokan.		*zoom* 0.2, width/height shift 0.2, *shear* 0.2, fill mode nearest, saturasi. Total data setelah augmentasi: 1.739 gambar. 5) Tiga model CNN diusulkan: - Model 1 (standar): 4 conv block, 9,8M parameter. - Model 2 (+BatchNorm) : 3 conv block + BN, 6,5M parameter. - Model 3		Model 2 kurang optimal (akurasi 82%). Model 3 unggul atas penelitian sebelumnya yang hanya 93% akurasi.	ke sistem/aplikasi nyata.	kematangan BUAH tomat (5 kelas kematangan); penelitian saat ini mengklasifikasikan PENYAKIT DAUN tomat. Penelitian terdahulu menggunakan CNN custom (scratch); penelitian saat ini menggunakan MobileNetV2 berbasis

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/peneliti an	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
				(+Dropout): 4 conv block + dropout, 2,4M parameter. 6) Hyperparameter: Adam, lr 0.001, *batch size* 16/32/64, *epoch* 50. 7) Evaluasi: *Confusion matrix*, akurasi, presisi, *recall*, *F1-Score*.				*transfer learning*. *Dataset* penelitian terdahulu dikumpulkan sendiri (500 gambar); penelitian saat ini menggunakan *dataset* PlantVillage. Penelitian terdahulu tidak mengembangkan aplikasi mobile; penelitian saat ini mengembangkan aplikasi

No	Judul, Penulis,Tahun	Masalah Penelitian	Tujuan Penelitian	Metodelogi Penelitian/Pemecahan masalah	Jumlah data/sample data/penelitian	Hasil Penelitian/Temuan Utama	Limitasi / Keterbatasan	Persamaan & Perbedaan dengan Penelitian Saat Ini
								Android.

2.3 Kerangka Teoritis *(Theoritical Framework)*

Kerangka acuan yang digunakan oleh peneliti yang merupakan hasil sintesa dari teori-teori yang relevan dan analisis penelitian-penelitian terdahulu, yang kemudian dibandingkan sebagi kemungkinan-kemungkinan solusi bagi penelitian yang dilakukan.



Gambar 2. 18 Kerangka Teoritis

Gambar 2.18 Kerangka teoritis yang menggambarkan hubungan antar konsep yang berfungsi sebagai landasan dalam penelitian. Pada penelitian ini berisi permasalahan, tujuan, proses, hasil, dan manfaat melalui pendekatan algoritma *Convolutional Neural Network* dengan arsitektur MobileNetV2 yang akan diimplementasikan pada aplikasi Android serta teori atau penelitian terdahulu yang relevan sebagai refrensi.