BAB II
LANDASAN TEORI

2.1 Kajian Literatur (*State Of The Art*)

Untuk menegaskan kebaruan (*novelty*) dan posisi unik dari penelitian ini, dilakukan penelusuran terhadap beberapa penelitian terdahulu yang memiliki keterkaitan topik. Tinjauan ini digunakan sebagai landasan untuk membandingkan metode dan hasil, sehingga posisi penelitian yang diusulkan tergambar dengan jelas. Rangkuman posisi penelitian dapat dilihat pada Tabel

Tabel 2. 1 *State Of The Art*

No	Penelitian, (Tahun)	Judul Penelitian	Metode	Object Penelitian	Perbandingan yang dijadikan alasan tinjau penelitian
1	Hamzah Raihan I. F., & Apriade Voutama (2023) DOI: https://doi.org/10.35457/antivirus.v17i1.2501	Pengujian *Black Box* Pada Aplikasi Database Perguruan Tinggi Dengan Teknik Equivalence Partitions	Pengujian *Black Box* dengan teknik Equivalence Partitions.	Aplikasi Database Perguruan Tinggi.	Penelitian terdahulu menguji sistem basis data teks. Penelitian ini menguji antarmuka aplikasi *full offline* berbasis AI (*Flutter*), mencakup fungsi kamera dan deteksi luring.
2	Pungky Rosalya Putri & Ronggo Alit (2024) DOI: https://doi.org/10.26740/jinacs.v6n03.p740-746	Klasifikasi Penyakit Diabetes Melitus Menggunakan Metode Support Vector Machine (SVM)	Algoritma Support Vector Machine (SVM).	Data Rekam Medis (Dataset Pima Indians Diabetes).	Penelitian terdahulu menggunakan algoritma *Machine Learning* (SVM) untuk klasifikasi data tabular (teks/angka). Penelitian ini menggunakan *Deep Learning* (*MobileNetV3*) untuk klasifikasi data citra visual (daun tomat).
3	Vaishnavi Pandey, et al. (2023)	Survey of *Accuracy* Prediction on the	Studi literatur (survei komparatif)	Dataset *PlantVillage* (Penyakit Tanaman).	Penelitian terdahulu sebatas studi komparasi teoretis model AI berukuran

No	Penelitian, (Tahun)	Judul Penelitian	Metode	Object Penelitian	Perbandingan yang dijadikan alasan tinjau penelitian
	DOI: https://doi.org/10.4108/eetiot.4578	*PlantVillage* Dataset using different ML techniques	terhadap berbagai arsitektur *Deep Learning* dan *Transfer Learning*.		besar (menemukan AlexNet terbaik). Penelitian ini mengimplementasikan model ringan (*MobileNetV3*) ke dalam aplikasi *full offline* yang fungsional untuk deteksi luring.
4	Mohammadreza Iman, et al. (2023) DOI: https://doi.org/10.3390/technologies11020040	A Review of Deep *Transfer Learning* and Recent Advancements	Studi literatur (Tinjauan komprehensif) terkait Deep *Transfer Learning* (DTL).	Konsep, taksonomi, dan algoritma Deep *Transfer Learning*.	Penelitian terdahulu sebatas tinjauan teoretis mengenai *Transfer Learning* secara umum. Penelitian ini mengimplementasikan *Transfer Learning* secara praktis pada *MobileNetV3* untuk deteksi penyakit tomat di perangkat *full offline*.
5	Rocco De Marco, et al. (2025) DOI : **https://doi.org/10.3390/robotics14050067**	*Real-time* Dolphin Whistle Detection on Raspberry Pi Zero 2 W with a TFLite *Convolutional Neural Network*	*Convolutional Neural Network* (*CNN*) teroptimasi dengan *TensorFlow Lite* (TFLite).	Deteksi suara lumba-lumba (pemrosesan sinyal audio/spektrogram).	Penelitian terdahulu mengimplementasikan model TFLite pada mikrokontroler (Raspberry Pi) untuk pemrosesan data audio. Penelitian ini mengimplementasikan model TFLite terintegrasi dengan *Flutter* pada *smartphone* untuk klasifikasi citra visual (penyakit daun tomat).
6	Annisa Tri Hidayati, et al. (2023) DOI : https://doi.org/10.55606/juprit.v2i4.2906	Perancangan Sistem Informasi Wirausaha Mahasiswa (Siwirma) Berbasis Web dengan Unified Modelling	*Software Development Life Cycle* (SDLC) model *Waterfall* dan pemodelan sistem UML.	Sistem Informasi Wirausaha Mahasiswa (Siwirma) berbasis Web.	Penelitian terdahulu menggunakan pemodelan UML untuk merancang sistem informasi manajemen berbasis *web*.

No	Penelitian, (Tahun)	Judul Penelitian	Metode	Object Penelitian	Perbandingan yang dijadikan alasan tinjau penelitian
		Languange (UML).			
7	Elizabeth Bjarnason, et al. (2023) DOI : https://doi.org/10.1007/s10664-023-10331-w	*An empirically based model of software Prototyping: a mapping study and a multi-case study*	Systematic mapping study dan multi-case study (wawancara & focus group).	Praktik software *Prototyping* pada rekayasa kebutuhan (*requirements engineering*).	penelitian terdahulu berfokus pada pembentukan model teoretis untuk mengevaluasi praktik *Prototyping* di perusahaan. Penelitian ini mengimplementasikan metode *Prototyping* secara praktis untuk membangun purwarupa aplikasi *full offline* cerdas pendeteksi penyakit tanaman.
8	Lobna M. Abouelmagd, et al. (2024) DOI : https://doi.org/10.1186/s13640-023-00618-9	*An optimized capsule neural networks for tomato leaf disease classification*	*Optimized Capsule Neural Network (CapsNet) dengan teknik Data Augmentation dan Adam optimizer.*	10 jenis penyakit pada daun tomat.	Penelitian terdahulu menggunakan arsitektur CapsNet yang cenderung kompleks dan membutuhkan sumber daya komputasi tinggi. Penelitian ini menggunakan arsitektur *MobileNetV3* yang teroptimasi TFLite sehingga jauh lebih ringan dan mampu dieksekusi secara luring pada perangkat *full offline* (ponsel pintar).
9	Raden Shalehuddin A., et al. (2024) DOI: https://doi.org/10.36040/jati.v8i1.8916	Implementasi SEBlock Pada Klasifikasi Citra Penyakit Mata Manusia Dengan Arsitektur *MobileNetV3-Small*	*Transfer Learning* dengan arsitektur *MobileNetV3-Small* SEBlock.	Citra Penyakit Mata Manusia.	Penelitian terdahulu mengimplementasikan arsitektur *MobileNetV3* pada ranah medis (penyakit mata). Penelitian ini mengadaptasi arsitektur *MobileNetV3* pada sektor pertanian (penyakit daun

No	Penelitian, (Tahun)	Judul Penelitian	Metode	Object Penelitian	Perbandingan yang dijadikan alasan tinjau penelitian
					tomat) dan mengintegrasikannya secara utuh ke dalam aplikasi *full offline* luring.
10	Tariku, G., dkk. (2024) DOI: **https://doi.org/10.3390/drones8110645**	*Advanced Image Preprocessing and Integrated Modeling for UAV Plant Image Classification*	Ekstraksi fitur VGG-16 & Klasifikasi SVM.	Citra tanaman (UAV/Drone).	Penelitian ini menggunakan metode *hybrid* (*CNN* dan SVM) untuk menekan beban komputasi, sebagai pembanding arsitektur dengan model *MobileNetV3* yang digunakan pada sistem deteksi penyakit tanaman saat ini.
11	Dewi, S., dkk. (2024) DOI : https://doi.org/10.56211/helloworld.v3i2.518	Klasifikasi Jenis Jerawat Berdasarkan Gambar Menggunakan Algoritma *CNN* (*Convolutional Neural Network*)	Algoritma *Convolutional Neural Network* (*CNN*).	Citra jenis jerawat.	Penelitian ini menjadi rujukan dasar implementasi *CNN* dalam mengekstrak fitur citra. Hal ini relevan sebagai tinjauan pustaka mengenai kemampuan dasar *CNN*, untuk kemudian dibandingkan dengan penggunaan arsitektur spesifik *MobileNetV3* pada sistem deteksi penyakit tanaman tomat.
12	Singarimbun, R. N., dkk. (2025) DOI : https://doi.org/10.32722/jap.v6i2.8125	Inovasi Teknologi Aplikasi *Mobile* Menggunakan Metode Pembelajaran Widget Dasar Pada Bahasa *Dart* Dalam *Framework Flutter*	Deskriptif kuantitatif (pendekatan practice-based learning).	Mahasiswa (pemahaman dan implementasi widget dasar *Flutter*).	Penelitian ini relevan sebagai rujukan literatur terkait implementasi antarmuka (UI) menggunakan *framework Flutter* dan bahasa *Dart*. Hal ini dapat dijadikan landasan teori untuk pengembangan antarmuka aplikasi

No	Penelitian, (Tahun)	Judul Penelitian	Metode	Object Penelitian	Perbandingan yang dijadikan alasan tinjau penelitian
					full offline pada sistem deteksi yang sedang kamu bangun.
13	Aung, S. T., dkk. (2024) DOI : **https://doi.org/10.3390/info15040191**	*A Study of Learning Environment for Initiating Flutter App Development Using Docker*	Lingkungan pengembangan terpadu (berbasis Docker) dengan pengujian praktik langsung.	Lingkungan pembelajaran pengembangan aplikasi *Flutter*.	Jurnal ini memperkuat alasan pemilihan *framework Flutter* dan *Dart* sebagai teknologi pengembangan aplikasi. Literatur ini dapat menjadi referensi pendukung yang menunjukkan efisiensi *Flutter* dalam membangun aplikasi dari satu basis kode (lintas platform), sejalan dengan teknologi aplikasi *full offline* yang sedang dikembangkan pada sistem.
14	Heydari, S., & Mahmoud, Q. H.(2025) DOI: **https://doi.org/10.3390/s25103191**	*Tiny Machine Learning and On-Device Inference: A Survey of Applications, Challenges, and Future Directions*	Tinjauan sistematis (Systematic Review) dengan panduan PRISMA.	Aplikasi TinyML dan inferensi on-device (pada perangkat).	Studi ini menjadi landasan teori pendukung tentang keunggulan inferensi AI langsung pada perangkat (*On-Device Inference*) dibandingkan komputasi awan

No	Penelitian, (Tahun)	Judul Penelitian	Metode	Object Penelitian	Perbandingan yang dijadikan alasan tinjau penelitian
					(*cloud*), seperti waktu respons yang lebih cepat dan efisiensi bandwidth. Argumen ini sangat kuat untuk membenarkan alasan penerapan model yang ringan seperti *MobileNetV3* secara lokal/luring pada aplikasi *full offline* pendeteksi penyakit tanaman tomat yang sedang dibangun.
15	Indra Suliwa, (2026)	Rancang Bangun Sistem Deteksi Penyakit Tanaman Tomat dan Rekomendasi Penanganan Menggunakan Model *DEEP LEARNING MOBILENE TV3*	Prototyping (SDLC), *Deep Learning* arsitektur *MobileNetV3 Small* (*Transfer Learning*), dan optimasi *TensorFlow Lite* (TFLite).	5 kelas citra daun tomat menggunakan *Hybrid dataset* (gabungan *PlantVillage* & kondisi nyata), diimplementasikan pada aplikasi Android full *offline* (*TomatoCare AI*).	Penelitian ini menjadi penyempurna dari penelitian-penelitian sebelumnya dengan menyelesaikan masalah *Domain shift* menggunakan *Hybrid dataset*, serta mengimplementasikan model yang ringan (*MobileNetV3*) ke dalam framework *Flutter* agar

No	Penelitian, (Tahun)	Judul Penelitian	Metode	Object Penelitian	Perbandingan yang dijadikan alasan tinjau penelitian
					proses klasifikasi dapat berjalan sepenuhnya secara *offline* (*On-Device Inference*).

2.2 Penelitian Terdahulu

Dalam upaya mengatasi permasalahan identifikasi penyakit pada daun tomat, berbagai metode dan strategi intervensi telah dilakukan oleh para peneliti sebelumnya melalui pendekatan komputasi. Secara teoretis, evolusi metode yang digunakan dapat dipetakan ke dalam beberapa strategi utama:

2.2.1 Intervensi dengan Machine Learning Konvensional

Sebagai upaya untuk mengatasi kelemahan dari besarnya komputasi arsitektur CNN pada umumnya, intervensi penelitian mulai diarahkan pada pemanfaatan arsitektur yang dirancang khusus untuk perangkat seluler (*mobile-based architecture*). Sebuah studi yang relevan mengeksplorasi perbandingan antara arsitektur *MobileNet* dan *NASNetMobile* dalam mengidentifikasi penyakit *Early Blight* dan *Late Blight* pada famili tanaman *Solanaceae* (kentang). Melalui berbagai skema pemisahan data latih dan uji, strategi pemanfaatan model berbobot ringan ini terbukti berhasil, di mana arsitektur *NASNetMobile* mampu mencapai tingkat akurasi hingga 90,96% pada skema data 90:10. Upaya penelitian ini memperkuat landasan teori bahwa implementasi Deep Learning tidak selalu harus bergantung pada infrastruktur komputasi yang berat, melainkan dapat dioptimalkan pada arsitektur ringan untuk menyelesaikan masalah klasifikasi gambar secara langsung di perangkat *mobile*.(Anwar Fuadi dan Aries Suharso, 2022).

2.2.2 Intervensi dengan *Deep Learning* Arsitektur Berat

Sebagai pembanding, strategi pemanfaatan *Deep Learning* berarsitektur berat juga terus dieksplorasi secara ekstensif. Sebuah studi berupaya mengatasi tantangan ekstraksi fitur penyakit daun tomat dengan melakukan intervensi menggunakan modifikasi arsitektur berskala besar, yaitu model *ResNet50-DPA*

(*Dual-Path Attention*). Secara teoretis, strategi ini berhasil mendongkrak akurasi dan mencegah hilangnya informasi fitur daun melalui penambahan lapisan *cascaded atrous convolution* serta mekanisme atensi. Kendati mampu memberikan hasil identifikasi yang sangat akurat, pencapaian tersebut menuntut konsekuensi tingginya kompleksitas model. Intervensi arsitektur yang sangat berat dan kompleks ini membutuhkan daya komputasi serta memori yang masif, sehingga implementasi praktisnya di lingkungan pertanian menjadi sangat terbatas dan kurang ideal untuk dieksekusi secara luring (*offline*) pada gawai bersumber daya rendah.(Liang & Jiang, 2023).

2.2.3 Intervensi dengan Data Laboratorium vs Data Nyata (*In-the-wild*)

Sebagai bukti nyata dari tantangan *domain shift* tersebut, sebuah studi spesifik berupaya mengatasi masalah degradasi performa model saat diuji silang antara data laboratorium dan lingkungan terbuka (*in-the-wild*). Penelitian tersebut mengusulkan intervensi *Unsupervised Domain Adaptation* menggunakan metode MSUN (*Multi-Representation Subdomain Adaptation Network*) untuk menekan penurunan akurasi pada pengenalan penyakit tanaman lintas domain. Walaupun menggunakan metode adaptasi yang kompleks, hasil eksperimen pada dataset daun tomat (*Tomato-Leaf-Diseases*) menunjukkan bahwa akurasi pengenalan penyakit secara silang tersebut terperosok hingga menyentuh angka 50,58%. Studi ini secara tegas mengonfirmasi teori bahwa model konvensional yang hanya diisolasi pada satu sumber data murni (laboratorium) akan kehilangan validitasnya saat berhadapan dengan lingkungan nyata yang bervariasi. Fakta akademis ini memperkuat justifikasi penggunaan strategi penggabungan data (*Hybrid dataset*) pada penelitian ini guna menanggulangi masalah variasi visual lapangan sejak fase pelatihan awal.(Wu et al., 2023)

2.3 Tinjauan Pustaka

Tinjauan pustaka merupakan bagian yang menguraikan berbagai konsep dasar, teori, dan teknologi pendukung yang menjadi landasan keilmuan dalam perancangan serta penyelesaian masalah pada penelitian ini. Bagian ini akan membahas secara komprehensif mengenai karakteristik tanaman tomat beserta penyakit yang menyerang bagian daun, dilanjutkan dengan penjabaran teknologi

kecerdasan buatan yang digunakan, meliputi konsep *Deep Learning*, arsitektur *Convolutional Neural Network* (*CNN*) khususnya varian *MobileNetV3*, serta optimasi model melalui *TensorFlow Lite* (TFLite) dan *Transfer Learning*. Selain itu, tinjauan ini juga memaparkan teknologi pengembangan sistem antarmuka berbasis *full offline* menggunakan kerangka kerja *Flutter* dan bahasa pemrograman *Dart*, konsep *On-Device Inference*, hingga metodologi rekayasa perangkat lunak yang mencakup metode *Prototyping*, pemodelan sistem menggunakan *Unified Modelling Language* (UML), dan teknik pengujian *Black Box*. Seluruh teori dasar ini saling berintegrasi untuk membangun kerangka berpikir yang solid dalam mengembangkan aplikasi klasifikasi penyakit daun tomat.

2.3.1 Tanaman Tomat dan Penyakit Daun

Tomat (*Solanum lycopersicum*) merupakan salah satu komoditas hortikultura unggulan yang memiliki nilai ekonomi tinggi dan dimanfaatkan secara luas dalam sektor pangan, farmasi, hingga kosmetik. Di Indonesia, kebutuhan akan konsumsi tomat terus meningkat, namun seringkali tidak diimbangi dengan tingkat produksi yang stabil.

Salah satu faktor utama penghambat produksi tomat adalah kerentanannya terhadap serangan penyakit tanaman. Infeksi penyakit, terutama yang menyerang bagian daun, dapat menyebabkan penurunan kualitas dan kuantitas produksi secara drastis, dengan potensi kerugian hasil panen mencapai 50% hingga 70%. (Rahmawati et al., 2025)

Healthy

Early Blight

Late Blight

Leaf Mold

Septoria Leaf Spot

Gambar 2. 1 Contoh 5 Kelas Daun Tomat

Sumber: Dataset *PlantVillage* (2015)

2.3.2 Penanganan Penyakit Daun Tomat

Penanganan penyakit daun tomat merupakan upaya yang dilakukan untuk mengurangi tingkat infeksi serta mencegah penyebaran penyakit pada tanaman. Menurut Panthee, Pandey, dan Paudel (2024), pengelolaan penyakit daun tomat dapat dilakukan melalui berbagai pendekatan, seperti sanitasi lahan, rotasi tanaman, penggunaan varietas tahan penyakit, pengaturan kelembapan lingkungan, serta aplikasi fungisida yang sesuai dengan jenis penyakit yang menyerang .(Panthee et al., 2024)

Selain itu, penerapan *Integrated Disease Management* (IDM) juga direkomendasikan karena mengombinasikan berbagai metode pengendalian secara terpadu, meliputi pendekatan kultur teknis, biologis, maupun kimiawi. Pendekatan tersebut bertujuan untuk menekan perkembangan penyakit sekaligus menjaga produktivitas tanaman secara berkelanjutan. Dalam penelitian ini, informasi penanganan penyakit digunakan sebagai basis pengetahuan untuk menghasilkan rekomendasi penanganan yang ditampilkan kepada pengguna setelah proses klasifikasi penyakit selesai dilakukan. (Panthee et al., 2024)

Tabel 2. 2 Klasifikasi Gejala Penyakit

Kelas	Karakteristik Gejala

Healthy	Daun berwarna hijau normal, tidak terdapat bercak, nekrosis, maupun perubahan bentuk daun.
Early Blight	Ditandai bercak cokelat berbentuk lingkaran konsentris (*bullseye*) yang umumnya muncul pada daun tua.
Late Blight	Muncul bercak cokelat kehitaman yang cepat meluas disertai gejala nekrosis pada jaringan daun.
Leaf Mold	Ditandai bercak kuning pada permukaan daun dan lapisan jamur berwarna hijau kecokelatan pada bagian bawah daun.
Septoria Leaf Spot	Memiliki bercak kecil berbentuk bulat berwarna abu-abu atau cokelat muda dengan tepi lebih gelap.

Sumber: Diadaptasi dari Panthee et al. (2024)

2.3.3 Klasifikasi dan Karakteristik Penyakit Daun Tomat

Secara spesifik, sistem deteksi pada penelitian ini difokuskan untuk mengenali lima kondisi utama pada daun tomat, yang diseleksi dari berbagai jenis penyakit umum yang menyerang komoditas ini. Berikut adalah penjabaran dari kelima kategori tersebut:

1. *Healthy* (Daun Sehat)

Gambar 2. 2 Citra Daun Tomat

Kondisi *Healthy* merupakan kelas kontrol yang mengindikasikan tanaman tidak terinfeksi patogen. Karakteristik visual daun sehat ditandai dengan pigmentasi warna hijau yang merata dan normal, tekstur jaringan

yang utuh, serta tidak ditemukannya bercak, klorosis, nekrosis, maupun perubahan bentuk *(deformitas*) pada morfologi daun.

2. *Early Blight* (Hawar Daun Awal)

Gambar 2. 3 Citra Daun Tomat *Early Blight*

Early Blight merupakan penyakit yang disebabkan oleh infeksi jamur Alternaria solani. Secara visual, patogen ini menghasilkan gejala berupa bercak cokelat kehitaman yang memiliki pola lingkaran konsentris menyerupai papan target (*bullseye*). Gejala ini umumnya muncul pertama kali pada daun-daun tua di bagian bawah tanaman sebelum akhirnya menyebar ke atas.

3. *Late Blight* (Hawar Daun Akhir)

Gambar 2. 4 Citra Daun Tomat *Late Blight*

Penyakit ini dipicu oleh patogen Phytophthora infestans yang sangat destruktif. Karakteristik utama dari *Late Blight* adalah munculnya bercak berwarna cokelat kehitaman yang tampak basah (lesi berair). Bercak ini menyebar dengan sangat cepat dan agresif, memicu gejala nekrosis

meluas yang dapat menghancurkan seluruh jaringan daun dalam waktu singkat.

4. *Leaf Mold* (Kapang Daun)

Gambar 2. 5 Citra Daun Tomat *Leaf Mold*

Leaf Mold disebabkan oleh jamur Passalora fulva (sebelumnya Fulvia fulva). Gejala spesifik dari penyakit ini dapat diidentifikasi melalui munculnya bercak berwarna kuning pucat yang tidak beraturan pada permukaan atas daun. Pada tahap lanjut, area bawah daun yang sejajar dengan bercak kuning tersebut akan ditumbuhi oleh lapisan massa jamur (kapang) yang berwarna hijau kecokelatan hingga keabu-abuan.

5. *Septoria Leaf Spot* (Bercak Daun Septoria)

Gambar 2. 6 Citra Daun Tomat *Septoria Leaf Spot*

Penyakit ini disebabkan oleh infeksi jamur Septoria lycopersici. Berbeda dengan hawar daun, karakteristik visual dari *Septoria Leaf Spot* berupa kemunculan banyak bercak berukuran kecil dan berbentuk bulat. Bagian tengah dari bercak ini umumnya berwarna abu-abu atau cokelat muda,

sementara bagian tepinya dikelilingi oleh batas berwarna lebih gelap (cokelat tua hingga kehitaman)

2.3.4 *Deep Learning* dan Pemrosesan Citra Digital (*Computer Vision*)

Pemrosesan Citra Digital (*Digital Image Processing*) memegang peranan krusial dalam mempersiapkan data visual sebelum dianalisis oleh sistem komputer. Kualitas citra mentah seringkali mengalami degradasi akibat berbagai faktor operasional dan lingkungan, seperti keterbatasan sensor kamera, resolusi yang rendah, hingga hilangnya informasi akibat pencahayaan atau bayangan. Oleh karena itu, tahapan prapemrosesan (*preprocessing*) citra yang mencakup peningkatan resolusi, penyesuaian kontras, serta kalibrasi keseimbangan warna (*white balance*) sangat dibutuhkan. Langkah prapemrosesan ini menjamin representasi warna yang akurat dan memastikan sistem menerima data masukan berkualitas tinggi.

Kualitas data masukan yang optimal tersebut akan berbanding lurus dengan kinerja model *Deep Learning* yang digunakan. *Deep Learning* adalah cabang dari kecerdasan buatan yang sangat andal dalam menangani data visual. Dalam klasifikasi citra, arsitektur *Deep Learning* seperti (*Convolutional Neural Network*) dipekerjakan sebagai pengekstraksi fitur (*feature extrActor*). Algoritma ini secara otomatis mempelajari pola dan karakteristik penting dari citra yang telah diprapemrosesan tanpa memerlukan rekayasa fitur secara manual. Kombinasi hibrida antara pemrosesan citra digital yang tepat dan arsitektur *Deep Learning* tidak hanya menghasilkan lonjakan akurasi klasifikasi yang signifikan, tetapi juga membuat penggunaan sumber daya komputasi menjadi lebih efisien.(Tariku et al., 2024)

2.3.5 *Convolutional Neural Networks* (*CNN*)

Dalam disiplin ilmu *Deep Learning*, *Convolutional Neural Networks (CNN)* diakui sebagai algoritma klasifikasi yang sangat tangguh untuk menangani analisis data citra digital. Arsitektur ini diimplementasikan sebagai pilar utama identifikasi visual karena keunggulannya dalam mengekstraksi representasi fitur hierarkis dari

suatu gambar secara otomatis. Berkat tahapan ekstraksi tersebut, komputasi sistem menjadi mampu untuk mendeteksi serta memahami pola-pola visual objek yang sangat kompleks dan terperinci. (Dewi et al., 2024)

Selain itu, efektivitas *CNN* dalam memproses citra pertanian juga dikonfirmasi oleh penelitian Thanjaivadivel et al. (2024), yang menemukan bahwa lapisan konvolusi pada arsitektur *CNN* mampu menganalisis secara mendalam properti morfologis dan karakteristik daun tanaman, yang meliputi elemen warna, intensitas, serta ukurannya, sehingga menghasilkan tingkat akurasi klasifikasi yang jauh lebih stabil dibandingkan metode ekstraksi fitur konvensional.(Gobinath et al., 2025)

2.3.5.1 Perumusan Matematis Convolutional Neural Network (*CNN*)

Secara matematis, operasi konvolusi pada jaringan saraf tiruan dilakukan melalui perkalian matriks antara peta fitur (feature map) masukan dengan matriks bobot (kernel / filter) (Gobinath et al., 2025), yang dirumuskan sebagai berikut:

$$\boldsymbol{z^l = a^{l-1} * W^l + b^l}$$

Keterangan:

a. z^l : Output dari layer konvolusi ke$-l$
b. a^{l-1}: Input fitur dari layer sebelumnya
c. W^l: Matriks bobot (Weight)
d. b^l: Nilai bias

Setelah proses konvolusi, data akan melewati tahap normalisasi dan aktivasi. Proses *Batch Normalization* diterapkan untuk mempercepat pelatihan dengan menormalkan distribusi input menggunakan rata-rata (*mean*) dan varians (*variance)* dari batch data:

$$\boldsymbol{X_{norm} = \frac{X - mean}{\sqrt{var + epsilon}}}$$

$$\boldsymbol{Y = \gamma \cdot X_{norm} +}$$

Di mana γ (*gamma*) dan β (*beta*) adalah parameter yang dipelajari selama pelatihan. Selanjutnya, fungsi aktivasi Rectified Linear Unit (ReLU) diaplikasikan untuk memperkenalkan sifat non-linear dengan mereduksi nilai negatif menjadi nol:

$$\boldsymbol{Y = \max(0, X)}$$

di mana X merepresentasikan tensor masukan dan Y adalah tensor keluaran.

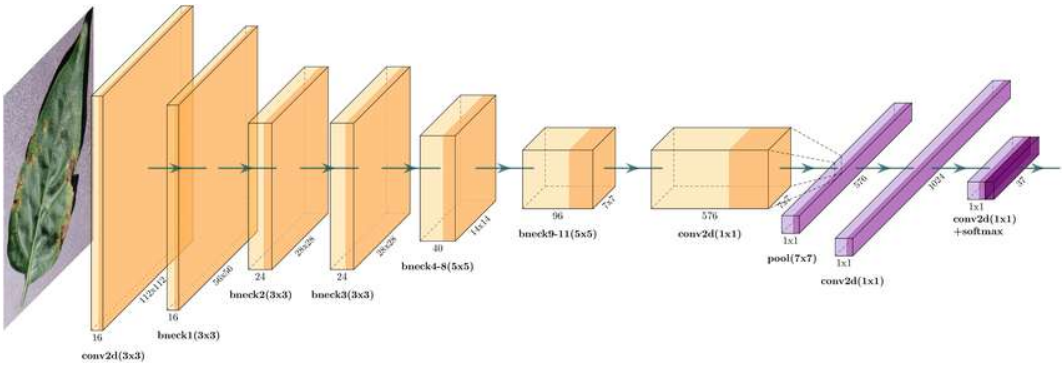
2.3.6 Arsitektur *MobileNetV3*

Arsitektur *MobileNetV3* merupakan pengembangan model *Deep Learning* yang dirancang khusus untuk mengoptimalkan kinerja pengenalan citra digital pada perangkat dengan spesifikasi komputasi yang terbatas (rendah). Pendekatan pada arsitektur ini sering kali dikombinasikan dengan metode *Transfer Learning* untuk mempercepat dan meningkatkan ketepatan proses pengenalan penyakit melalui gambar.

Salah satu inovasi krusial dalam arsitektur ini (khususnya pada varian *MobileNetV3-Small*) adalah implementasi *Squeeze-and-Excitation* Block (SEBlock). Penggunaan metode tersebut memungkinkan model untuk menghasilkan keluaran prediksi yang maksimal meskipun dijalankan pada lingkungan perangkat keras yang minim sumber daya. Melalui tahap pelatihan, arsitektur *MobileNetV3* terbukti menunjukkan performa komputasi yang sangat stabil. Model ini memiliki rekam jejak pengujian yang sangat baik dalam membaca dan mengklasifikasi ribuan kumpulan data (dataset) gambar berskala besar, sehingga mampu mencetak nilai akurasi yang sangat memuaskan. Karakteristik inilah yang menjadikannya sangat ideal untuk diimplementasikan sebagai inti dari sistem deteksi berbasis perangkat bergerak. (Albawani et al., 2024)

Keandalan arsitektur ini juga sejalan dengan hasil studi komprehensif yang dilakukan oleh (Gobinath et al., 2025). Dalam penelitiannya, disimpulkan bahwa efisiensi komputasi yang tinggi pada *MobileNetV3* dicapai berkat kombinasi modul depthwise separable convolutions dan *Squeeze-and-Excitation* (SE) yang terbukti ampuh dalam memangkas jumlah parameter komputasi tanpa mengorbankan kemampuan ekstraksi fitur. Selain itu, implementasi fungsi aktivasi *hard-swish* (*h-*

swish) pada arsitektur ini secara signifikan mampu meningkatkan stabilitas pelatihan serta mempercepat konvergensi model. Karakteristik arsitektur yang sangat ringan tersebut menjamin waktu inferensi yang cepat dan menekan biaya komputasi, menjadikannya solusi komputasi yang sangat ideal untuk diimplementasikan pada perangkat seluler maupun lingkungan *Edge Computing* yang memiliki keterbatasan memori dan sumber daya.



Gambar 2. 7 Aristektur *MobileNetV3 Small*

Sumber: Howard et al. (2019)

2.3.6.1 Perumusan Matematis Arsitektur *MobileNetV3*

Arsitektur *MobileNetV3* mengandalkan teknik *Depthwise Separable Convolution* (DS-*CNN*) yang memisahkan proses konvolusi spasial dan kanal (channelwise) menjadi dua operasi terpisah untuk memangkas jumlah parameter secara signifikan (Dhanalaxmi et al., 2025). Tahap pertama adalah *Depthwise Convolution* yang mengaplikasikan satu filter 3×3 untuk setiap kanal input:

$$\boldsymbol{Z_i(x,y) = \sum_{a=1}^{3}\sum_{b=1}^{3} X_i(x+a-2, y+b-2) \cdot D_i(a,b,i)}$$

di mana $Z_i(x,y)$ adalah output feature map untuk kanal i, X_i adalah input feature map, dan D_i melambangkan filter depthwise pada posisi (a,b). Proses ini kemudian diikuti oleh *Pointwise Convolution* menggunakan filter 1×1 untuk menggabungkan kembali fitur-fitur dari berbagai kanal.

Selain efisiensi konvolusi, *MobileNetV3* menggunakan fungsi aktivasi h-swish (hard-swish) sebagai pengganti fungsi swish standar untuk menekan biaya komputasi, yang dirumuskan secara matematis sebagai:

$$\boldsymbol{hswish(x) = x \cdot \frac{ReLU6(x+3)}{6}}$$

Fungsi linier piece-wise ini memungkinkan komputasi berjalan sangat ringan tanpa menghilangkan properti representasi non-linear dari jaringan.

2.3.7 Konsep *On-Device Inference*

Pertumbuhan implementasi kecerdasan buatan telah memicu peningkatan kebutuhan akan pemrosesan data dan komputasi inferensi. Selama ini, solusi inferensi tradisional yang berbasis komputasi awan (*cloud-based inference*) sering kali digunakan. Namun, pendekatan ini terbukti tidak memadai untuk aplikasi yang menuntut waktu respons hampir seketika (*near-instantaneous response times*). Sebagai solusi, konsep inferensi pada perangkat (*On-Device Inference*) atau yang sering dikaitkan dengan *Tiny Machine Learning* (TinyML) hadir sebagai alternatif utama.

Pendekatan *On-Device Inference* menjadi sangat krusial pada aplikasi di mana penundaan transmisi data (*transmission delays*) membuat sistem tradisional tidak praktis untuk digunakan. Konsep ini menawarkan keunggulan yang signifikan dibandingkan inferensi berbasis *cloud*, terutama ketika diimplementasikan di lingkungan dengan keterbatasan lebar pita jaringan (bandwidth constraints) seperti area perkebunan. Dengan memproses model langsung di dalam perangkat keras lokal, *On-Device Inference* memiliki rekam jejak keandalan yang kuat untuk penerapan di dunia nyata *(real-world usability*) serta mampu memberikan waktu respons yang sangat cepat. (Heydari & Mahmoud, 2025)

2.3.8 *Framework Flutter*

Framework Flutter yang dioperasikan bersama dengan bahasa pemrograman *Dart* memungkinkan pengembang untuk membangun aplikasi lintas platform, baik untuk perangkat seluler maupun web, secara efisien hanya melalui

satu basis kode (). Keunggulan utama dari kerangka kerja ini terletak pada kemampuannya *single codebase* untuk melakukan konversi secara efisien menjadi kode asli (*native codes*) saat diimplementasikan sebagai aplikasi seluler.(Aung et al., 2024)

Selain mendukung pengembangan aplikasi lintas platform, *Flutter* juga menyediakan berbagai widget yang telah terintegrasi sehingga memudahkan proses perancangan antarmuka pengguna (*User Interface*) yang responsif dan konsisten pada berbagai ukuran perangkat. Pada penelitian ini, *Flutter* digunakan sebagai *framework* utama dalam pengembangan aplikasi deteksi penyakit tanaman tomat karena memiliki kompatibilitas yang baik dengan *TensorFlow Lite* , sehingga model *Deep Learning* yang telah dikonversi dapat diintegrasikan dan dijalankan secara langsung pada perangkat Android. Dengan pendekatan tersebut, proses deteksi penyakit dapat dilakukan secara lokal (*On-Device Inference*) tanpa memerlukan koneksi internet maupun server eksternal.

2.3.9 *Dart*

Dart merupakan bahasa pemrograman utama yang beroperasi secara terintegrasi di dalam kerangka kerja (*framework*) *Flutter* untuk mendukung pengembangan aplikasi bergerak (*full offline* application development). Dalam praktiknya, bahasa pemrograman *Dart* digunakan untuk menyusun, mengimplementasikan, dan mengelola konsep dasar *widget* pada antarmuka aplikasi. Penggunaan bahasa *Dart* dalam pengembangan perangkat lunak dinilai sangat adaptif dan memiliki tingkat relevansi yang tinggi terhadap pemenuhan kebutuhan industri di era digital yang berbasis aplikasi.(Singarimbun et al., 2025)

2.3.10 *SharedPreferences* dan Penyimpanan Lokal (*Local Storage*)

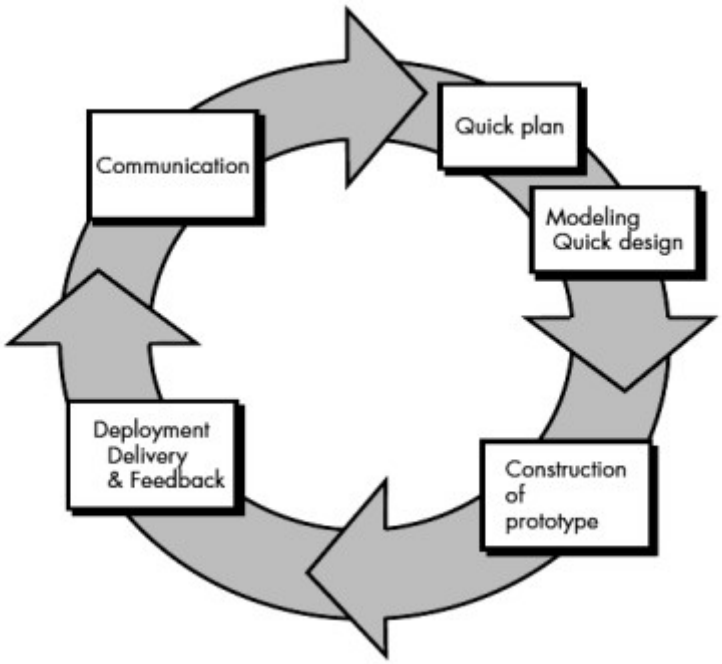
SharedPreferences merupakan antarmuka (interface) penyimpanan data lokal yang digunakan secara luas pada pengembangan aplikasi bergerak, termasuk pada kerangka kerja *Flutter*. Berbeda dengan sistem manajemen basis data relasional (RDBMS) seperti *SQLite* yang menggunakan struktur tabel kompleks, *SharedPreferences* beroperasi menggunakan mekanisme *Key-Value Pair*

(pasangan Kunci-Nilai) untuk menyimpan data primitif secara asinkron dan persisten di dalam memori internal perangkat.

Menurut penelitian yang dilakukan oleh (Wahyudi et al, 2025) terkait perancangan aplikasi inventori berbasis *Flutter*, integrasi *SharedPreferences* memegang peran penting sebagai sarana penyimpanan data konfigurasi lokal secara persisten (seperti status sesi atau *cache*). Penyimpanan lokal ini memastikan bahwa informasi esensial tetap dapat diakses dengan cepat meskipun koneksi jaringan tidak stabil atau aplikasi telah ditutup. Lebih lanjut, pendekatan ini sangat mendukung arsitektur *client-server* maupun aplikasi luring (*offline*) karena meminimalisir ketergantungan perangkat terhadap peladen (server) dan menjaga kontinuitas akses antarmuka dengan latensi yang sangat rendah.

2.3.11 Metode *Prototyping*

Metode *Prototyping* (purwarupa) merupakan praktik yang telah mapan dan diakui secara luas dalam proses desain produk serta perancangan antarmuka pengguna (*User Interface* design). Dalam konteks rekayasa perangkat lunak, metode ini secara aktif diimplementasikan sebagai instrumen utama dalam praktik rekayasa kebutuhan (*requirements engineering*), terutama pada lingkungan pengembangan perangkat lunak yang tangkas (*Agile development*), (Bjarnason et al., 2023).



Gambar 2. 8 Diagram Metode Prototyping

Sumber: Diadaptasi dari Bjarnason et al.,(2023)

Tahapan standar dalam siklus hidup pengembangan sistem prototype berdasarkan diagram di atas meliputi:

1. Komunikasi (*Communication*)

   Tahap awal di mana pengembang dan pengguna bertemu untuk mendefinisikan format, fungsionalitas, serta mengidentifikasi permasalahan yang menjadi dasar pengembangan sistem secara global.
2. Perencanaan Cepat (*Quick Plan*)

   Proses penyusunan rencana iterasi secara singkat yang berfokus pada penyelesaian masalah dan pemenuhan kebutuhan yang telah disepakati pada tahap komunikasi.
3. Pemodelan dan Perancangan Cepat (*Modeling Quick Design*)

   Tahap perancangan arsitektur kasar dan antarmuka sistem (seperti UML dan wireframe) yang berfokus pada representasi visual dari perangkat lunak agar dapat dievaluasi dengan cepat.
4. Pembentukan Purwarupa (*Construction of Prototype*)

Tahap penerjemahan hasil perancangan ke dalam bentuk kode program (coding) untuk menghasilkan sampel produk awal yang dapat dioperasikan.

5. Penyerahan dan Umpan Balik (*Deployment, Delivery, and Feedback*)
   Proses penyerahan purwarupa kepada pengguna untuk diuji secara langsung. Umpan balik (*feedback*) dari pengguna digunakan sebagai bahan evaluasi untuk menyempurnakan purwarupa pada iterasi selanjutnya hingga sistem siap dioperasikan secara penuh.

2.3.11.1 Perbandingan dengan Metode Pengembangan Lain

Untuk menentukan model pengembangan perangkat lunak yang paling adaptif terhadap integrasi model *Deep Learning* ke dalam aplikasi berbasis *mobile*, dilakukan komparasi teoretis antara metode *Waterfall*, *Agile* (Scrum), dan *Prototyping* (Senarath, 2021), yang disajikan pada tabel 2

Tabel 2. 3 Perbandingan dengan Metode Pengembangan Lain

No	Karakteristik	Metode *Waterfall*	Metode *Agile*	Metode *Prototyping*
1	Alur Kerja	Mengalir secara linear dan sekuensial, di mana suatu fase hanya dapat dimulai jika fase sebelumnya telah selesai.	Dibangun secara bertahap melalui iterasi pendek (sprint) yang berdurasi antara 1 hingga 4 minggu.	Eksploratif dan interaktif melalui perancangan sampel awal aplikasi yang dibangun, diuji, dan disempurnakan secara berulang.
2	Keterlibatan Pengguna	Rendah, karena pengguna hanya dapat melihat produk perangkat lunak secara utuh di akhir siklus pengembangan.	Sangat tinggi, melalui kolaborasi mandiri tim lintas fungsi dan komunikasi yang berkesinambungan dengan pengguna.	Sangat tinggi, terutama saat pengguna memberikan evaluasi dan umpan balik secara langsung terhadap purwarupa yang diajukan
3	Manajemen Risiko	Memiliki risiko dan ketidakpastian yang tinggi,	Risiko dapat ditekan melalui penyesuaian di setiap siklus	Risiko sangat rendah, karena kesalahan atau fungsi yang

No	Karakteristik	Metode *Waterfall*	Metode *Agile*	Metode *Prototyping*
		karena kesalahan desain atau integrasi sering kali baru terdeteksi di bagian paling akhir proses.	iterasi yang singkat dan responsif.	membingungkan dapat dideteksi dan diperbaiki jauh lebih awal melalui purwarupa.
4	Fleksibilitas Perubahan	Sangat kaku dan tidak cocok untuk proyek yang kebutuhannya berisiko tinggi mengalami perubahan di tengah jalan.	Sangat adaptif dalam menyelaraskan proses pengembangan dengan perubahan kebutuhan bisnis yang dinamis.	Sangat optimal digunakan pada skenario di mana pengguna belum dapat merumuskan spesifikasi kebutuhannya secara rinci sejak awal.

Sumber : Diadaptasi dari Senarath (2021)

2.3.11.2 Alasan Pemilihan Metode *Prototyping*

Berdasarkan matriks komparasi pada Tabel 2.1, metode *Waterfall* dinilai kurang relevan untuk diterapkan pada penelitian ini. Metode *Waterfall* memiliki karakteristik alur yang kaku, di mana perangkat lunak yang dapat berfungsi baru dihasilkan pada tahap akhir siklus hidup pengembangan. Hal ini memicu risiko dan ketidakpastian yang tinggi, mengingat perubahan atau kesalahan integrasi teknologi yang terdeteksi di fase pengujian akan sangat sulit dan mahal untuk diperbaiki. Di sisi lain, metode *Agile* (Scrum) memberikan fleksibilitas tinggi untuk merespons perubahan kebutuhan yang dinamis melalui siklus pengembangan singkat. Meskipun demikian, metode ini menuntut pembentukan tim lintas fungsi yang sepenuhnya mandiri serta kolaborasi harian yang sangat padat, sehingga kurang proporsional untuk skala pengerjaan skripsi mandiri.(Senarath, 2021)

Oleh karena itu, metode *Prototyping* dipilih sebagai kerangka kerja yang paling rasional dan presisi. Metode *Prototyping* memungkinkan perancangan sampel sistem awal yang dapat segera dievaluasi oleh pengguna untuk memahami fungsionalitas aplikasi yang sesungguhnya. Keterlibatan pengguna yang intensif di awal proses ini memfasilitasi deteksi kelemahan operasional dan cacat fungsi jauh

lebih dini, sehingga menghemat waktu dan biaya pengembangan secara signifikan. Siklus iteratif ini akan terus berulang untuk menyempurnakan performa model AI dan antarmuka *mobile* berdasarkan umpan balik pengguna, hingga menghasilkan produk akhir yang secara utuh memenuhi kebutuhan penyelesaian masalah.

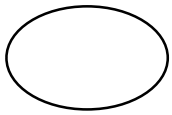
2.3.12 *Unified Modelling Language* (UML)

Dalam perancangan sistem informasi maupun aplikasi, fase desain merupakan tahapan krusial yang tidak mungkin diabaikan, mengingat perannya sebagai elemen tak terpisahkan dari *Software Development Life Cycle* (SDLC). Guna mencapai eksekusi rekayasa perangkat lunak yang ideal, seluruh rancangan sistem menuntut adanya dokumentasi teknis yang sistematis. Salah satu standar visualisasi yang diimplementasikan secara luas untuk mengakomodasi kebutuhan dokumentasi tersebut adalah *Unified Modelling Language* (UML)(Hidayati et al., 2023).

A. *Use Case Diagram*

   Use Case Diagram berfungsi untuk merepresentasikan aspek dinamis dari sebuah sistem. Secara spesifik, diagram ini digunakan untuk memperoleh dan memetakan kebutuhan sistem (*system requirements*), termasuk melihat bagaimana pengaruh dari pihak internal maupun eksternal yang berinteraksi dengan sistem tersebut. (Gobinath et al., 2025)

Tabel 2. 4 Simbol *Use Case Diagram*

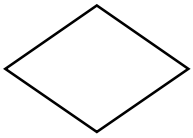
No	Simbol	Nama Simbol	Deskripsi
1		*Actor*	Mewakili peran orang, sistem yang lain, atau alat ketika berkomunikasi dengan usecase
2		*Use Case*	Abstraksi dan Interaksi antar sisstem dan aktor
3		*Association*	Abstraksi dari penghubung antara aktor dan *Use Case*

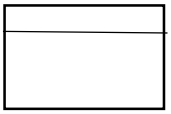
Sumber : Diadaptasi dari Journal (Gobinath et al., 2025)

B. Activity Diagram

Activity Diagram merupakan diagram alir (flowchart) yang merepresentasikan aliran data dari satu aktivitas ke aktivitas lainnya di dalam sistem. Diagram ini mendeskripsikan secara rinci proses operasi yang melibatkan interaksi pengguna, di mana interaksi tersebut dijembatani oleh antarmuka pengguna (*User Interface*).(Gobinath et al., 2025)

Tabel 2. 5 Simbol Activity Diagram

No	Simbol	Nama Simbol	Deskripsi
1		Status Awal	Diagram aktivitas pasti memiliki status awal
2		Aktivitas	Aktivitas yang diawali sistem biasanya diawalai dengan kata kerja
3		Percabangan/ *Decison*	Percabangan yang dimana ada pilihan aktivitas yang lebih dari satu

No	Simbol	Nama Simbol	Deskripsi
4		Penggabungan/ Join	Penggabungan yang mana lebih dari satu aktivitas lalu digabungkan menjadi satu
4		Status Akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
5		*Swimlane*	Swimlane memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

Sumber : Diadaptasi dari Journal (Gobinath et al., 2025)

C. *Sequence Diagram*

Sequence Diagram adalah diagram yang memvisualisasikan alur interaksi antar objek atau komponen pada suatu proses atau sistem. Isi dari *Sequence Diagram* harus sama dengan *Use Case Diagram* dan *Class Diagram*.

Tabel 2. 6 Simbol *Sequence Diagram*

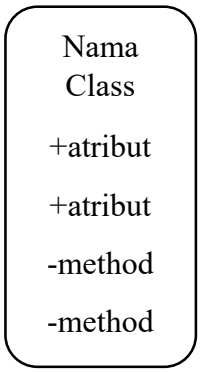
No	Simbol	Nama Simbol	Keterangan
1		*Actor*	Segala sesuatu yang berinteraksi dengan sistem aplikasi *Mobile*
2		Garis Hidup	Menyatakan kehidupan suatu objek
3	<<create>>	Pesan tipe *create*	Objek yang lain, arah panah mengarah pada objek.
4	1.namamethod()	Pesan tipe *call*	Menyatakan suatu objek memanggil operasi yang ada pada objek lain.
5	1: masukan	Pesan Tipe *send*	Menyatakan suatu objek memanggil operasi yang ada pada objek lain.

Sumber: Journal Sistem informasi berbasi Web (Malius et al., 2021)

D. *Class Diagram*

Simbol Diagram Kelas UML adalah notasi standar yang digunakan untuk merepresentasikan struktur dan hubungan kelas dalam sistem berorientasi objek. Simbol-simbol ini secara visual mengkomunikasikan bagaimana kelas, atribut, metode, dan hubungan berinteraksi dalam desain perangkat lunak. Simbol-simbol ini berfungsi sebagai bahasa universal bagi pengembang, analis, dan perancang, membantu tim memodelkan sistem dengan jelas tanpa ambiguitas.

Tabel 2. 7 *Class Diagram*

No	Nama	Simbol	Keterangan
1	Dependency		Penggunaan dependency digunakan untuk menunjukkan operasi pada suatu class yang menggunakan class yang lain.
2	Class		Class adalah blok-blok pembangun pada pemrograman berorientasi objek. Class digambarkan sebagai sebuah kotak yang terbagi atas 3 bagian. Bagian atas adalah bagian nama dari class. Bagian tengah mendefinisikan atribut class. Bagian akhir mendefinisikan method-method

Sumber: Journal Sistem informasi berbasi Web (Malius et al., 2021)

2.3.13 *TensorFlow Lite* (TFLite)

Sebagai framework turunan, *TensorFlow Lite* (TFLite) diciptakan secara spesifik untuk memfasilitasi delegasi dan eksekusi model *Machine Learning* pada perangkat berspesifikasi komputasi minim (*resource-constrained settings*) semacam *smartphone*. Keunggulan paling menonjol dari TFLite adalah kapabilitasnya dalam melakukan optimalisasi arsitektur terhadap model kecerdasan buatan orisinal yang sebelumnya dibangun melalui *TensorFlow*.(Marco et al., 2025)

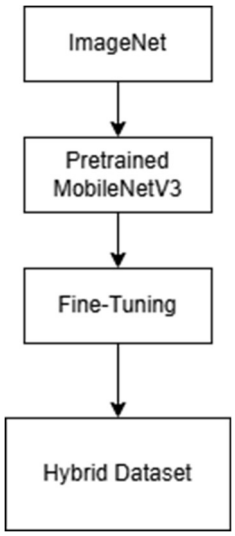
Lebih jauh lagi, integrasi *Machine Learning* pada ponsel pintar mengalami percepatan yang signifikan berkat adanya TFLite. Mengingat fungsinya sebagai varian yang lebih ringan (*lightweight*) dari *TensorFlow* utama, framework ini memang difokuskan agar proses komputasi berjalan lebih sigap pada *mobile* devices maupun *embedded devices*. *Model Deep Learning* hasil pelatihan nantinya bisa langsung dioperasikan dengan memanfaatkan interpreter TFLite yang sudah tersemat (*embedded*) di dalam basis aplikasi. Pendekatan ini menjadi opsi yang sangat ideal untuk mengimplementasikan sistem pakar pertanian secara *offline*, di mana memori gawai milik petani dapat memproses data secara lokal tanpa perlu khawatir akan adanya latensi dari koneksi internet..(Gobinath et al., 2025)

2.3.14 *Transfer Learning*

Transfer Learning merupakan sebuah pendekatan dalam pengembangan *Deep Learning* yang bertujuan untuk menggunakan kembali pengetahuan (*knowledge*) yang telah dipelajari oleh sebuah model dari suatu tugas atau data sumber (*source task*), untuk diterapkan pada tugas atau data target (*target task*). Pendekatan ini secara efektif menjadi solusi atas dua kendala utama dalam pelatihan jaringan saraf tiruan tradisional, yaitu tingginya ketergantungan terhadap ketersediaan data latih berlabel dalam jumlah masif serta besarnya beban komputasi selama proses pelatihan.

Melalui penerapan *Transfer Learning*, ketergantungan terhadap pengumpulan data latih yang ekstensif dapat ditekan dan durasi serta biaya pelatihan model dapat diminimalisir secara drastis. Penurunan beban komputasi ini menjadikan teknik tersebut sangat layak dan efisien untuk diimplementasikan pada perangkat di ujung jaringan (*edge devices*) yang memiliki keterbatasan sumber daya. Hal ini sangat mendukung skenario pemindahan model ke dalam ponsel pintar tanpa

mengorbankan kualitas pengetahuan dari model pra-latih (pre-trained model) aslinya, (Iman & Arabnia, 2023).



Gambar 2. 9 Transfer Learing

Sumber: Penulis (2026)

2.3.15 *PlantVillage* Dataset

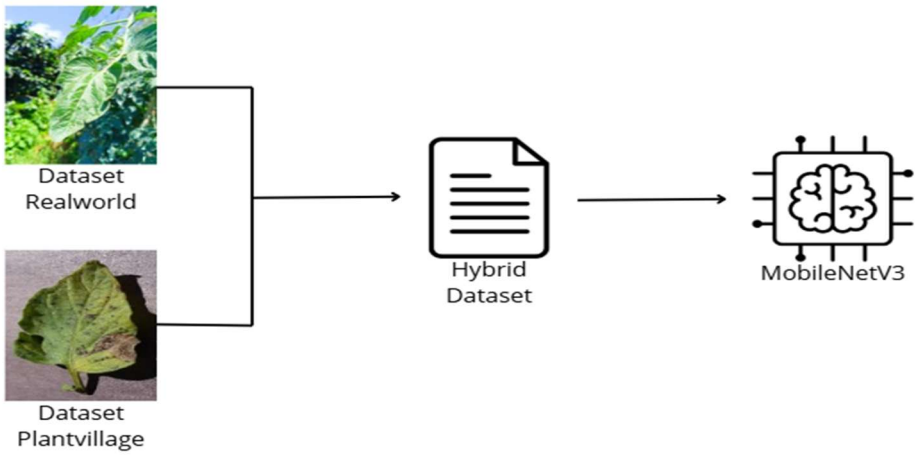
Dataset *PlantVillage* hadir sebagai salah satu basis data publik berisikan citra digital yang paling banyak diadopsi sebagai standar evaluasi (benchmark) utama dalam penelitian klasifikasi penyakit tanaman. Kumpulan data ini mencakup dokumentasi karakteristik visual citra daun dengan beragam jenis patogen penyakit. Ketersediaan data yang masif dan terstruktur pada Dataset *PlantVillage* telah terbukti mampu mendukung proses pelatihan dan evaluasi berbagai arsitektur model *Deep Learning* modern untuk menghasilkan tingkat akurasi deteksi yang tinggi, bahkan efektivitasnya sering kali melampaui kemampuan pengamatan visual manusia maupun metode pemrosesan citra konvensional.(Pandey et al., 2024)

2.3.16 Hybrid dataset

Pentingnya pendekatan hibrida dalam penyusunan data pelatihan juga dikonfirmasi oleh penelitian (An et al., 2023). Berdasarkan penelitian tersebut, objek di dunia nyata sering kali memiliki karakteristik visual yang kompleks dan memuat berbagai kombinasi atribut sekaligus, sehingga sulit didefinisikan jika hanya menggunakan satu kriteria data yang kaku. Dengan menerapkan kerangka

kerja *Hybrid dataset* pada metode supervised learning, model kecerdasan buatan dapat dilatih secara lebih efisien untuk memahami kombinasi fitur visual yang bervariasi. Pendekatan ini secara empiris terbukti krusial untuk meningkatkan akurasi kemampuan komputer dalam mengenali serta mengklasifikasikan objek saat model diimplementasikan ke dalam aplikasi berbasis data sesungguhnya.

Pada penelitian ini, *Hybrid dataset* dibentuk dari penggabungan dataset *PlantVillage* dan dataset citra daun tomat kondisi nyata yang diperoleh dari area pertanian di Pangalengan. Penggabungan kedua dataset tersebut bertujuan untuk meningkatkan kemampuan generalisasi model *MobileNetV3* sehingga mampu mendeteksi penyakit daun tomat pada berbagai kondisi lingkungan dan karakteristik citra yang berbeda.



Gambar 2. 10 *Hybrid Dataset*

Sumber: Penulis (2026)

2.3.17 *Confusion Matrix* dan Metrik Evaluasi

Confusion Matrix merupakan sebuah representasi matriks yang memvisualisasikan perbandingan secara detail antara kelas prediksi yang dihasilkan oleh model komputasi dengan kelas aktual pada data pengujian (testing data). Melalui nilai-nilai parameter yang dihasilkan dari matriks tersebut, evaluasi kinerja dapat dijabarkan lebih lanjut ke dalam beberapa metrik spesifik, antara lain tingkat akurasi (*Accuracy*), presisi (*Precision*), sensitivitas atau daya ingat (*Recall*), serta *F1-Score* . Penggunaan keseluruhan metrik evaluasi ini merupakan standar validasi

yang bertujuan untuk memastikan ketepatan prediksi model dan meminimalisir kesalahan klasifikasi, sehingga model dijamin kelayakannya sebelum diimplementasikan sebagai sistem pendukung keputusan.(Putri & Alit, 2024)

2.3.17.1 Metrik Evaluasi Performa

Menurut (Automatisés et al., 2024),penilaian performa pada model klasifikasi *Deep Learning* dianalisis melalui indikator Quality of Service (QoS). Standar pengukuran ini diturunkan langsung dari empat komponen dasar yang terdapat dalam *Confusion Matrix*, yakni *True Positive (TP), True Negative (TN), False Positive (FP),* serta *False Negative (FN).* Adapun formulasi matematis untuk mengkalkulasi metrik-metrik evaluasi tersebut dijabarkan sebagai berikut:

A. Akurasi (*Accuracy*)

Akurasi mengukur proporsi keseluruhan dari prediksi yang benar (positif maupun negatif) terhadap total seluruh data yang diuji.

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

B. Presisi (*Precision*)

Presisi menghitung rasio jumlah observasi positif yang diprediksi secara tepat dibandingkan dengan keseluruhan hasil yang diprediksi sebagai kelas positif.

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

C. Sensitivitas (*Recall* / Sensitivity)

Recall atau sensitivitas mengukur kemampuan model dalam mengidentifikasi secara benar seluruh observasi positif yang aktual.

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

D. *F1-Score* (*F-Measure*)

F-Measure adalah metrik yang mencari nilai keseimbangan (rata-rata harmonik) antara Presisi dan *Recall*, yang sangat berguna ketika terdapat ketidakseimbangan jumlah data pada kelas pengujian.

$$\boldsymbol{F\text{-}Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall}}$$

2.3.18 Domain shift

Domain shift merupakan tantangan fundamental dalam visi komputer, di mana distribusi data pada tahap pelatihan memiliki perbedaan karakteristik yang signifikan dengan distribusi data pada tahap implementasi lapangan (*target domain*). Fenomena ini umumnya dipicu oleh variasi proses akuisisi data, perbedaan pencahayaan, latar belakang, hingga variasi karakteristik objek yang tidak terwakili dalam dataset pelatihan (Mehrtens et al., 2023) .Menurut penelitian Mehrtens et al. (2023), kerentanan model *Deep Neural Networks* (DNN) terhadap *Domain shift* ini dapat mengakibatkan penurunan performa klasifikasi yang drastis, sehingga membatasi kemampuan sistem dalam memberikan estimasi ketidakpastian (*predictive uncertainty*) yang dapat dipercaya saat digunakan di lingkungan baru yang tidak terprediksi. Oleh karena itu, kemampuan model untuk mengenali variasi distribusi data adalah prasyarat krusial agar sistem pendukung keputusan, terutama yang berbasis citra medis maupun pertanian, dapat beroperasi secara reliabel dan aman..

2.3.19 *Data Augmentation*

Pentingnya peran *Data Augmentation* dalam meningkatkan performa model *Deep Learning* juga diperkuat oleh penelitian (Alomar & Aysel, 2023). Berdasarkan hasil surveinya, model *Deep Neural Networks* seperti *CNN* memiliki ketergantungan yang tinggi terhadap ketersediaan data dalam skala besar agar dapat mencapai generalisasi yang optimal. Tanpa volume data yang memadai, model cenderung mengalami *overfitting*, di mana model hanya menghafal data pelatihan dan kehilangan kemampuan untuk mengenali pola pada data baru. Teknik *Data Augmentation* menawarkan solusi efektif untuk mengatasi kendala keterbatasan data dengan cara menciptakan variasi sampel baru secara artifisial, yang terbukti secara konsisten mampu meningkatkan akurasi klasifikasi dan memperkuat ketahanan model terhadap skenario data dunia nyata.

Tabel 2. 8 *Data Augmentation*

Teknik	Fungsi
RandomFlip	Membalik citra
RandomRotation	Rotasi citra
RandomZoom	Zoom citra
RandomContras	Mengubah kontras
RandomTranslation	Menggeser posisi objek

Sumber: Penulis (2026)

2.3.20 Pengujian *Black Box*

Fase pengujian perangkat lunak memegang peranan vital di dalam siklus pengembangan sistem guna memvalidasi mutu dari sebuah aplikasi sebelum akhirnya didistribusikan kepada pengguna akhir. Dari sekian banyak teknik evaluasi yang ada, pengujian *Black Box* merupakan salah satu pendekatan yang paling sering diaplikasikan. Karakteristik utama dari metode ini adalah peninjauannya yang berfokus secara eksklusif pada kesesuaian output dan fungsionalitas operasional sistem, sehingga proses pemeriksaannya sama sekali tidak menuntut akses ke dalam struktur source code internal di balik aplikasi tersebut..(Raihan et al., 2023).