

BAB II

KAJIAN PUSTAKA

A. Kajian Teoritis

1. *Artificial Intelligence*

Artificial intelligence merupakan cabang ilmu komputer yang berupaya membuat mesin mampu meniru kecerdasan manusia, seperti kemampuan belajar, menalar, dan mengambil keputusan. Dalam perkembangannya, AI melahirkan berbagai sub-bidang, di antaranya *machine learning* dan *deep learning*, yang menjadi landasan utama dalam penelitian klasifikasi citra penyakit tanaman ini (Alzubaidi *et al.*, 2021).

Kecerdasan buatan modern tidak lagi sekadar menjalankan aturan (*rule-based*) yang ditetapkan secara eksplisit, melainkan mampu mengenali pola, beradaptasi terhadap situasi baru, dan meningkatkan kinerjanya melalui proses pembelajaran dari data. Kemampuan inilah yang membuat AI banyak diterapkan pada berbagai bidang, mulai dari pengenalan suara, pemrosesan bahasa alami, hingga visi komputer (*computer vision*).

Dalam konteks pertanian, penerapan kecerdasan buatan membuka peluang besar untuk membantu petani mengambil keputusan secara cepat dan tepat. Salah satu penerapan yang paling relevan adalah visi komputer untuk mengenali objek pada citra, misalnya mengidentifikasi gejala penyakit pada daun tanaman. Dengan pendekatan berbasis data, sistem cerdas dapat mempelajari ciri visual penyakit dari ribuan contoh citra

sehingga mampu melakukan klasifikasi secara otomatis tanpa bergantung pada penilaian manual yang subjektif.

Berdasarkan uraian di atas dapat disimpulkan bahwa kecerdasan buatan merupakan fondasi utama yang menaungi *machine learning* dan *deep learning*. Dalam penelitian ini, kecerdasan buatan menjadi landasan konseptual bagi pembangunan sistem klasifikasi penyakit tanaman tomat dan kentang yang mampu belajar dari data citra dan mengambil keputusan secara otomatis.

2. *Machine Learning*

Machine learning adalah sub-bidang kecerdasan buatan yang memungkinkan komputer belajar dari data tanpa diprogram secara eksplisit. Proses pembelajaran mesin pada umumnya dibagi menjadi dua langkah, yaitu melatih model menggunakan sejumlah besar data sehingga model memperoleh hubungan optimal antar variabel, kemudian memasukkan data baru ke model yang telah dilatih untuk memprediksi keluarannya. Pada kasus klasifikasi citra, pendekatan *machine learning* klasik seperti *Support Vector Machine* (SVM) memerlukan ekstraksi fitur secara manual, sedangkan *deep learning* mampu mempelajari fitur secara otomatis dari data (Alzubaidi *et al.*, 2021; Liew *et al.*, 2021)

Secara umum, *machine learning* dikelompokkan menjadi tiga paradigma utama, yaitu *supervised learning*, *unsupervised learning*, dan *reinforcement learning*. Pada *supervised learning*, model dilatih menggunakan data yang telah diberi label sehingga dapat memetakan

masuk ke keluaran yang benar; pendekatan inilah yang digunakan dalam klasifikasi citra penyakit tanaman, di mana setiap citra daun telah dilabeli dengan jenis penyakitnya. *Unsupervised learning* bekerja pada data tanpa label untuk menemukan pola tersembunyi, sedangkan *reinforcement learning* mempelajari kebijakan tindakan melalui mekanisme *reward* dan *punishment*.

Keberhasilan sebuah model *machine learning* sangat dipengaruhi oleh kualitas dan kuantitas data serta proses ekstraksi fitur. Pada pendekatan *machine learning* klasik, fitur seperti warna, tekstur, dan bentuk harus dirancang serta diekstraksi secara manual oleh peneliti, sehingga membutuhkan pengetahuan domain yang mendalam dan rentan terhadap kesalahan. Keterbatasan ini mendorong berkembangnya *deep learning* yang mampu mempelajari representasi fitur secara otomatis dan bertingkat langsung dari data mentah.

Berdasarkan uraian di atas dapat disimpulkan bahwa *machine learning* memungkinkan komputer belajar dari data untuk melakukan prediksi. Dalam penelitian ini, pendekatan yang dipilih adalah *supervised learning* berbasis *deep learning* karena mampu mempelajari fitur citra penyakit secara otomatis sehingga lebih akurat dibandingkan *machine learning* klasik yang bergantung pada ekstraksi fitur manual.

3. *Deep Learning*

Deep Learning (DL) adalah sub-bidang pembelajaran mesin yang didasarkan pada pembelajaran beberapa *level* representasi dengan

membangun hierarki fitur, di mana *level* yang lebih tinggi ditentukan dari *level* yang lebih rendah. Struktur DL memperluas jaringan saraf tradisional (*neural network*) dengan menambahkan lebih banyak Lapisan tersembunyi (*hidden layer*) di antara Lapisan *input* dan *output* untuk memodelkan hubungan yang lebih *kompleks* dan *nonlinier*. Konsep ini menarik minat para peneliti karena kinerjanya yang baik dalam banyak masalah aplikasi analisis citra, seperti *denoising* citra, segmentasi, registrasi, dan klasifikasi (Alzubaidi *et al.*, 2021).

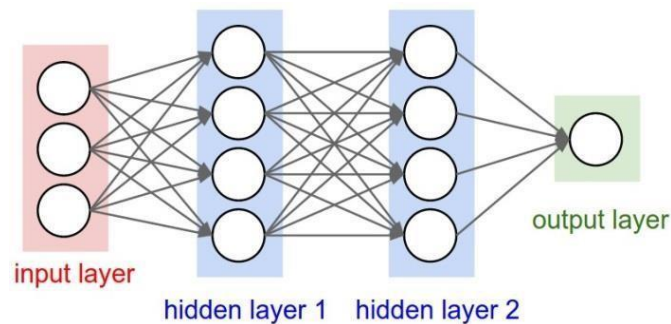
Keunggulan utama *deep learning* terletak pada kemampuannya melakukan *feature learning* secara otomatis. Melalui lapisan-lapisan tersembunyi yang bertingkat, jaringan mampu mengekstraksi representasi mulai dari fitur tingkat rendah, seperti tepi dan tekstur, hingga fitur tingkat tinggi yang lebih abstrak, seperti bentuk dan pola objek. Proses ini menghilangkan kebutuhan akan rekayasa fitur manual yang menjadi kendala pada *machine learning* klasik.

Perkembangan *deep learning* didorong oleh tiga faktor utama, yaitu ketersediaan data berskala besar, peningkatan kemampuan komputasi terutama melalui *Graphics Processing Unit* (GPU), serta kemajuan algoritma pelatihan jaringan yang semakin dalam. Dalam bidang analisis citra, arsitektur *deep learning* yang paling banyak digunakan dan terbukti unggul adalah *Convolutional Neural Network* (CNN) karena dirancang khusus untuk mempertahankan dan memanfaatkan informasi spasial pada citra.

Berdasarkan uraian di atas dapat disimpulkan bahwa *deep learning* merupakan pengembangan jaringan saraf tiruan dengan banyak lapisan yang mampu mempelajari fitur secara hierarkis. Dalam penelitian ini, *deep learning* menjadi metode inti yang diwujudkan melalui arsitektur CNN untuk mengklasifikasikan penyakit tanaman tomat dan kentang dari citra daun.

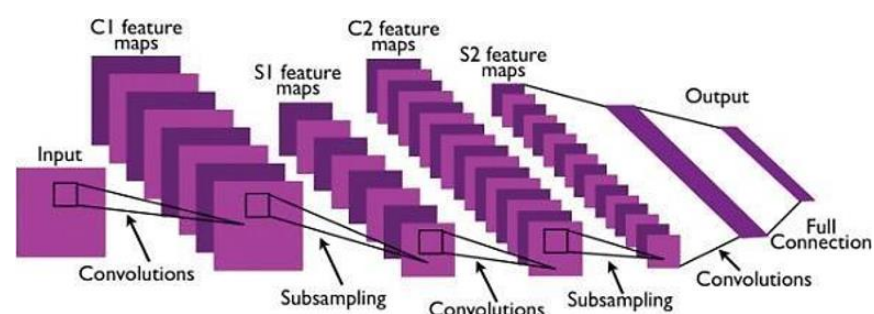
4. *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah pengembangan dari *Multilayer Perceptron* (MLP) yang didesain untuk mengolah data dua dimensi. CNN termasuk dalam jenis *Deep Neural Network* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. Pada kasus klasifikasi citra, MLP kurang sesuai digunakan karena tidak menyimpan informasi spasial dari data citra dan menganggap setiap *piksel* sebagai fitur yang independen, sehingga menghasilkan hasil yang kurang baik. CNN pertama kali dikembangkan dengan nama *NeoCognitron* oleh Kunihiko Fukushima, kemudian dimatangkan oleh Yann LeCun melalui model LeNet pada penelitian pengenalan angka dan tulisan tangan. Pada tahun 2012, Alex Krizhevsky dengan penerapan CNN-nya berhasil menjuarai kompetisi *ImageNet Large Scale Visual Recognition Challenge* 2012, yang menjadi momen pembuktian bahwa metode *deep learning*, khususnya CNN, mampu mengungguli metode *machine learning* lain seperti SVM pada kasus klasifikasi objek pada citra (Chen *et al.*, 2021).



Gambar 2. 1 Arsitektur MLP Sederhana (Chen *et al.*, 2021)

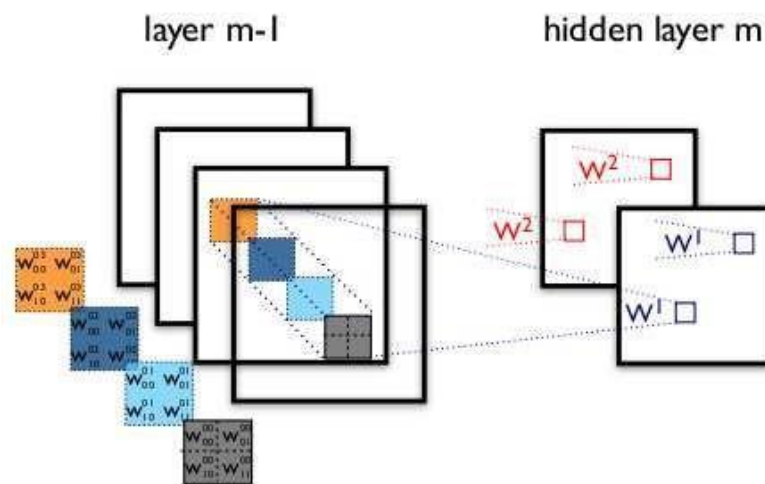
Cara kerja CNN tidak jauh berbeda dengan MLP. Perbedaan utamanya adalah neuron pada CNN berukuran dua dimensi, sedangkan neuron pada MLP hanya berukuran satu dimensi. Pada CNN, operasi linear yang digunakan adalah operasi *konvolusi*, dengan parameter bobot berbentuk empat dimensi yang merupakan kumpulan *kernel konvolusi*. CNN bekerja melalui serangkaian Lapisan komputasi yang mengekstraksi fitur dari citra secara hierarkis, dari fitur sederhana seperti tepi dan tekstur hingga fitur *kompleks* seperti bentuk dan pola objek. Secara umum, arsitektur CNN tersusun atas beberapa jenis Lapisan utama sebagai berikut.



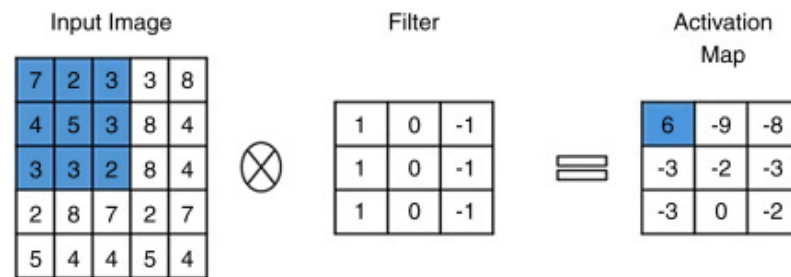
Gambar 2. 2 Arsitektur *Convolutional Neural Network* (Chen *et al.*, 2021)

a. *Convolution Layer*

Convolutional layer adalah blok bangunan utama CNN. Lapisan ini berisi sekumpulan filter (*kernel*) yang parameternya dipelajari selama pelatihan, dengan ukuran filter biasanya lebih kecil daripada gambar sebenarnya. Setiap filter melintasi tinggi dan lebar gambar, dan perkalian titik (*dot product*) antara setiap elemen filter dengan masukan dihitung pada setiap posisi spasial untuk menghasilkan peta aktivasi (*feature map*). Lapisan *konvolusi* awal menangkap fitur tingkat rendah seperti garis dan tepi, sedangkan lapisan selanjutnya mengekstraksi fitur tingkat tinggi seperti bentuk dan objek tertentu (Mostafa & Wu, 2021).

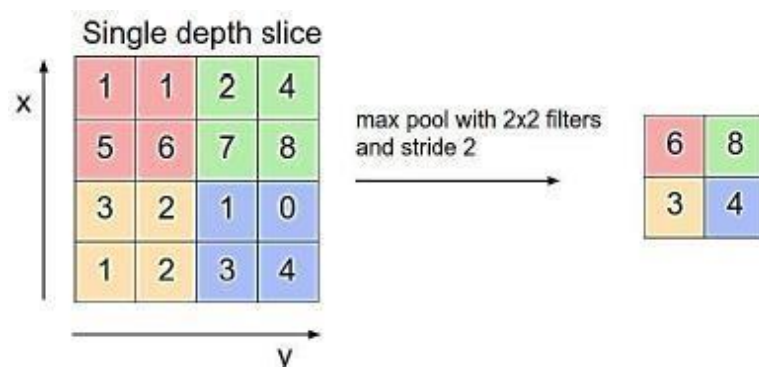


Gambar 2. 3 *Convolution Layer* (Chen et al., 2021)

Gambar 2. 4 Operasi *Konvolusi* (Mostafa & Wu, 2021)

b. *Pooling Layer*

Tujuan utama *Pooling Layer* adalah mengurangi jumlah parameter dari tensor *input* sehingga membantu mengurangi *overfitting*, mengekstraksi fitur representatif, dan mengurangi beban komputasi. Terdapat dua jenis *pooling* yang umum digunakan, yaitu *Max Pooling* dan *Average Pooling*. Pada *Max Pooling*, kernel berukuran $n \times n$ dipindahkan melintasi *matriks* dan untuk setiap posisi diambil nilai maksimalnya, sedangkan pada *Average Pooling* diambil nilai rata-ratanya. Proses ini menurunkan dimensi tinggi dan lebar citra (*downsampling*), tetapi jumlah channel (kedalaman) tetap sama (Chen *et al.*, 2021).

Gambar 2. 5 Operasi *Max Pooling* (Chen *et al.*, 2021)

c. *Batch Normalization*

Batch Normalization adalah pendekatan untuk melatih jaringan saraf yang sangat dalam dengan membakukan (standardisasi) *input* ke suatu Lapisan untuk setiap mini-batch. Hasilnya, proses pembelajaran menjadi lebih stabil dan pelatihan jaringan dalam membutuhkan jumlah *epoch* yang jauh lebih sedikit (Chang & Lai, 2024).

d. *Dropout Layer*

Dropout menyediakan metode komputasi murah untuk meregularisasi model dan mencegah *overfitting*. Selama pelatihan, sejumlah unit (*neuron*) yang dipilih secara acak dihilangkan sementara dari jaringan beserta seluruh koneksi masuk dan keluarnya. Mekanisme ini secara efektif mencegah *overfitting* sekaligus seperti menggabungkan banyak arsitektur jaringan saraf yang berbeda secara efisien (de Silva & Kunz, 2023).

e. *Flatten Layer*

Flatten layer meruntuhkan dimensi spasial dari *feature map* yang telah terbentuk sambil mempertahankan dimensi *channel*, kemudian mengubahnya menjadi vektor satu dimensi. Vektor inilah yang selanjutnya diumpankan ke lapisan *fully connected* (dikenal juga sebagai *dense layer*) (Paymode & Malode, 2022).

f. Fungsi Aktivasi

Fungsi aktivasi merupakan operasi matematika yang diaplikasikan pada sinyal keluaran untuk menentukan apakah suatu neuron aktif atau tidak. Fungsi aktivasi yang sering dipakai dalam CNN antara lain $\tanh()$, $ReLU$ (*Rectified Linear Unit*), *Sigmoid*, dan *Softmax* (Alzubaidi *et al.*, 2021). Fungsi $ReLU$ menghasilkan nilai 0 jika nilai masukannya negatif, dan menghasilkan nilai masukan itu sendiri jika positif. Persamaan $ReLU$ ditunjukkan pada persamaan (1):

$$f(x) = \max(0, x) \quad \dots\dots\dots (1)$$

Aktivasi *Softmax* umumnya diterapkan pada lapisan terakhir jaringan saraf. *Softmax* berguna untuk mengubah keluaran *neural network* menjadi distribusi probabilitas, sehingga cocok untuk klasifikasi multi-kelas seperti pada penelitian ini (9 kelas). Persamaan *Softmax* ditunjukkan pada persamaan (2):

$$f(x_i) = e^{(x_i)} / \sum_j e^{(x_j)}, \text{ untuk } i = 1, 2, 3, \dots, k \quad \dots\dots\dots (2)$$

g. Fully Connected Layer

Fully Connected Layer adalah Lapisan yang biasa digunakan dalam MLP untuk melakukan perubahan dimensi data agar dapat dilakukan pengklasifikasian secara linear. Seluruh neuron pada *convolution layer* terlebih dahulu diubah menjadi data satu dimensi sebelum diinput ke *fully connected layer*. Lapisan ini hanya dapat ditempatkan pada akhir jaringan karena dapat menyebabkan hilangnya informasi spasial (Chen *et al.*, 2021).

h. Loss Function

Fungsi loss yang umum digunakan pada klasifikasi citra adalah *cross-entropy loss (log loss)*. Fungsi ini mengukur kinerja model klasifikasi melalui perhitungan probabilitas antara 0 dan 1; nilai *cross-entropy loss* akan meningkat jika probabilitas yang diprediksikan menyimpang dari label sebenarnya (Liew *et al.*, 2021). Untuk masalah multi-kelas seperti pada penelitian ini, digunakan *categorical cross-entropy loss* yang mengukur perbedaan antara distribusi probabilitas prediksi dan label kebenaran (Yeung *et al.*, 2022).

Berdasarkan uraian di atas dapat disimpulkan bahwa *Convolutional Neural Network* merupakan arsitektur *deep learning* yang tersusun atas rangkaian Lapisan *konvolusi*, *pooling*, dan Lapisan terhubung penuh yang bekerja secara hierarkis dalam mengekstraksi dan mengklasifikasikan fitur citra. Kemampuan CNN dalam mempertahankan informasi spasial dan mempelajari fitur secara otomatis menjadikannya metode yang paling tepat digunakan dalam penelitian klasifikasi penyakit tanaman tomat dan kentang berbasis citra daun.

5. MobileNetV2 dan Transfer Learning

MobileNetV2 adalah arsitektur CNN yang ringan namun berperforma tinggi, dirancang khusus untuk *deployment* pada perangkat dengan sumber daya terbatas seperti *smartphone Android*. *MobileNetV2* dipilih karena memiliki jumlah parameter yang relatif sedikit, mendukung

ukuran *input* citra standar 224 x 224 *piksel*, sehingga cocok dijalankan secara langsung (*on-device*) dengan *response time* yang cepat (Gulzar, 2023).

Transfer Learning adalah proses pemindahan pengetahuan (*knowledge*) dari pelatihan sebelumnya untuk digunakan pada penelitian baru, sehingga waktu pelatihan menjadi lebih cepat dibandingkan melatih jaringan dari awal yang membutuhkan waktu komputasi lama. Cara kerja *transfer learning* adalah dengan mempertahankan parameter pada *layer-layer* awal dari model yang telah dilatih pada *dataset* besar (seperti *ImageNet*), menghilangkan *layer* terakhir, lalu melatih ulang *layer* terakhir tersebut menggunakan *dataset* baru (Gulzar, 2023; Chang & Lai, 2024). Pendekatan ini sangat efektif ketika jumlah data pelatihan terbatas, karena memanfaatkan representasi fitur umum yang telah dipelajari sebelumnya. Dalam penelitian ini, *MobileNetV2* digunakan dengan pendekatan *transfer learning*: bobot *pre-trained* dari *ImageNet* dijaga (*frozen*) pada *layer* awal, sementara *layer* atas dilatih ulang menggunakan *dataset* penyakit daun tomat dan kentang (Paymode & Malode, 2022).

Arsitektur *MobileNetV2* memperkenalkan dua inovasi utama, yaitu *inverted residual structure* dan *linear bottleneck*, yang secara signifikan mengurangi jumlah operasi dan parameter tanpa mengorbankan akurasi. Penggunaan *depthwise separable convolution* memecah operasi *konvolusi* standar menjadi dua tahap yang lebih efisien sehingga beban komputasi menjadi jauh lebih ringan. Karakteristik inilah yang membuat *MobileNetV2*

sangat sesuai dijalankan secara *on-device* pada *smartphone* dengan keterbatasan daya dan memori (Gulzar, 2023).

Berdasarkan uraian di atas dapat disimpulkan bahwa kombinasi *MobileNetV2* dan *transfer learning* memungkinkan pembangunan model klasifikasi yang ringan, cepat, dan akurat meskipun dilatih dengan jumlah data terbatas. Dalam penelitian ini, pendekatan tersebut dipilih agar model CNN dapat dijalankan langsung pada perangkat *Android* petani secara efisien tanpa memerlukan sumber daya komputasi yang besar.

6. *TensorFlow dan TensorFlow Lite*

TensorFlow merupakan sistem matematika simbolis berbasis pemrograman aliran data (*dataflow*) yang banyak digunakan dalam realisasi algoritma *machine learning*. *TensorFlow* 2.0 telah terintegrasi dengan Keras sehingga mempercepat pengembangan model. Library ini digunakan dalam penelitian ini untuk membangun dan melatih model CNN *MobileNetV2* (Hu & Zhang, 2021).

TensorFlow Lite (Tflite) adalah versi ringan dari *TensorFlow* yang dioptimalkan untuk inferensi pada perangkat *mobile* dengan sumber daya komputasi terbatas. Model CNN yang telah dilatih dikonversi ke format *tflite* menggunakan *TfliteConverter*; proses ini melakukan kuantisasi bobot sehingga ukuran file berkurang dan kecepatan inferensi meningkat tanpa menurunkan akurasi secara signifikan. Integrasi *Tflite* ke dalam aplikasi *Android* dilakukan menggunakan *Tflite Interpreter API*,

memungkinkan proses klasifikasi penyakit berjalan secara *offline* langsung di perangkat pengguna tanpa memerlukan koneksi internet.

Ekosistem *TensorFlow* menyediakan rangkaian alat lengkap yang mencakup proses pembangunan, pelatihan, evaluasi, hingga penerapan model. Integrasinya dengan Keras memungkinkan model didefinisikan secara ringkas melalui *API* tingkat tinggi, sementara dukungan komputasi GPU mempercepat proses pelatihan pada *dataset* citra berukuran besar. Setelah model dilatih, *TensorFlow* menyediakan mekanisme *konversi* ke format yang lebih ringan untuk keperluan penerapan pada perangkat *mobile* (Hu & Zhang, 2021).

Berdasarkan uraian di atas dapat disimpulkan bahwa *TensorFlow* digunakan untuk membangun dan melatih model, sedangkan *TensorFlow Lite* berperan mengonversi model tersebut agar dapat berjalan efisien pada perangkat *mobile*. Dalam penelitian ini, kedua pustaka tersebut menjadi tulang punggung implementasi model CNN sehingga proses klasifikasi penyakit dapat dilakukan secara *offline* langsung di perangkat *Android* pengguna.

7. Metrik Evaluasi Model

Evaluasi performa model klasifikasi pada penelitian ini menggunakan *confusion matrix* beserta metrik turunannya. *Confusion matrix* membandingkan nilai aktual dan nilai prediksi, di mana elemen diagonal menunjukkan hasil yang diklasifikasikan dengan benar. Dari *confusion matrix* diperoleh empat nilai dasar, yaitu *True Positive* (TP),

False Positive (FP), *True Negative* (TN), dan *False Negative* (FN) (Attallah, 2023). Berdasarkan nilai-nilai tersebut dihitung metrik berikut:

Akurasi mengukur rasio prediksi yang benar terhadap seluruh prediksi, persamaan (3):

$$Accuracy = (TP + TN) / (P + N) \quad \dots\dots\dots (3)$$

Recall mengukur proporsi kasus positif aktual yang teridentifikasi dengan benar, persamaan (4):

$$Recall = TP / (TP + FN) \quad \dots\dots\dots (4)$$

Precision mengukur proporsi prediksi positif yang benar, persamaan (5):

$$Precision = TP / (TP + FP) \quad \dots\dots\dots (5)$$

F1-Score merupakan rata-rata harmonik dari *precision* dan *recall*, persamaan (6):

$$F1-Score = 2 \times (Precision \times Recall) / (Precision + Recall) \quad \dots\dots\dots (6)$$

Keempat metrik tersebut memberikan sudut pandang yang saling melengkapi dalam menilai kinerja model. Akurasi memberikan gambaran umum performa, namun dapat menyesatkan apabila jumlah data antar kelas tidak seimbang. *Precision* penting ketika kesalahan positif palsu (*false positive*) berdampak besar, sedangkan *recall* menjadi krusial ketika kasus positif yang terlewat (*false negative*) harus diminimalkan. F1-score menyeimbangkan *precision* dan *recall* sehingga cocok digunakan pada

kondisi data yang tidak seimbang, seperti pada *dataset* penyakit tanaman yang jumlah citranya bervariasi antar kelas (Attallah, 2023).

Berdasarkan uraian di atas dapat disimpulkan bahwa evaluasi model tidak cukup hanya mengandalkan nilai akurasi, melainkan perlu ditinjau secara menyeluruh melalui *confusion matrix* beserta *precision*, *recall*, dan *F1-score*. Dalam penelitian ini, keempat metrik tersebut digunakan bersama-sama untuk mengukur kinerja model CNN secara objektif pada seluruh kelas penyakit tomat dan kentang.

8. Tanaman Tomat

Tanaman tomat (*Solanum lycopersicum*) merupakan salah satu komoditas hortikultura penting yang memiliki nilai ekonomi tinggi dan banyak dibudidayakan di Indonesia. Tomat sangat rentan terhadap serangan berbagai penyakit yang sulit dibedakan secara visual, sehingga menyebabkan keterlambatan penanganan dan kerugian ekonomi bagi petani (Edoazani *et al.*, 2025). Beberapa penyakit utama yang menyerang tanaman tomat antara lain:

- a. *Bacterial Spot (Xanthomonas vesicatoria)* - penyakit bakteri yang menyebabkan bercak kecil bersudut, hitam basah pada daun.
- b. *Leaf Mold (Passalora fulva)* - jamur penyebab bercak kuning di permukaan atas daun dan Lapisan jamur abu-zaitun di bawah daun.
- c. *Tomato Mosaic Virus* - virus yang menyebabkan pola mosaik kuning-hijau, daun keriting, dan pertumbuhan tanaman terhambat.

Secara *morfologi*, tanaman tomat memiliki daun majemuk berwarna hijau yang menjadi bagian paling sering menunjukkan gejala awal serangan penyakit. Gejala visual seperti perubahan warna, munculnya bercak, serta perubahan bentuk dan tekstur daun umumnya menjadi indikator utama keberadaan penyakit. Namun, gejala antar penyakit kerap terlihat serupa sehingga menyulitkan petani awam membedakannya hanya melalui pengamatan mata (Edoazani *et al.*, 2025).

Keterlambatan mengidentifikasi jenis penyakit dapat mengakibatkan penanganan yang keliru, penyebaran penyakit yang semakin luas, serta penurunan hasil panen secara signifikan. Oleh karena itu, dibutuhkan sistem bantu yang mampu mengenali jenis penyakit tomat secara cepat dan akurat berdasarkan citra daun sehingga petani dapat mengambil tindakan pengendalian yang tepat sejak dini.

Berdasarkan uraian di atas dapat disimpulkan bahwa tanaman tomat memiliki nilai ekonomi tinggi namun rentan terhadap berbagai penyakit dengan gejala visual yang mirip. Dalam penelitian ini, tiga penyakit utama tomat, yaitu *Bacterial Spot*, *Leaf Mold*, dan *Tomato Mosaic Virus*, dijadikan objek klasifikasi bersama kondisi daun sehat.

9. Tanaman Kentang

Tanaman kentang (*Solanum tuberosum*) merupakan komoditas pangan strategis yang penting bagi ketahanan pangan nasional dan sangat rentan terhadap serangan penyakit yang dapat menurunkan kualitas dan

kuantitas produksi. Identifikasi dini menjadi kunci keberhasilan budidaya.

Beberapa penyakit utama pada kentang antara lain:

- a. *Early Blight (Alternaria solani)* - bercak coklat berbentuk cincin konsentris pada daun tua, umumnya menyerang lebih awal di musim.
- b. *Late Blight (Phytophthora infestans)* - penyakit busuk daun yang sangat merusak, dengan bercak coklat berair yang cepat meluas.
- c. *Common Scab (Streptomyces scabies)* - penyakit pada umbi kentang dengan bercak menonjol seperti kudis pada permukaan umbi yang menurunkan nilai jual.

Kentang merupakan salah satu sumber karbohidrat utama yang banyak dibudidayakan di dataran tinggi Indonesia. Sebagaimana tomat, tanaman kentang tergolong famili *Solanaceae* sehingga memiliki kerentanan terhadap beberapa patogen yang serupa, terutama cendawan dan bakteri yang menyerang daun serta umbi. Gejala penyakit pada kentang umumnya muncul pada daun berupa bercak dengan pola dan warna tertentu, yang apabila tidak ditangani dapat menjalar dan merusak seluruh tanaman.

Serangan penyakit seperti *Late Blight* bahkan tercatat dalam sejarah sebagai penyebab kegagalan panen kentang berskala besar sehingga deteksi dini menjadi sangat penting. Kesulitan membedakan jenis penyakit secara visual, ditambah keterbatasan akses petani terhadap tenaga ahli, memperkuat kebutuhan akan sistem klasifikasi otomatis berbasis citra yang dapat digunakan secara mandiri di lapangan.

Berdasarkan uraian di atas dapat disimpulkan bahwa tanaman kentang merupakan komoditas strategis yang rentan terhadap penyakit daun dan umbi. Dalam penelitian ini, tiga penyakit utama kentang, yaitu *Early Blight*, *Late Blight*, dan *Common Scab*, dijadikan objek klasifikasi bersama kondisi daun sehat.

10. Pengembangan Aplikasi *Android*

Android merupakan platform *mobile* dengan penetrasi yang sangat luas di masyarakat, sehingga menjadi pilihan ideal untuk menjembatani teknologi klasifikasi citra dengan kebutuhan praktis petani di lapangan. Aplikasi pada penelitian ini dikembangkan menggunakan *Android Studio* dengan bahasa pemrograman *Java*. *Java* merupakan bahasa pemrograman berorientasi objek yang masih banyak digunakan dalam pengembangan aplikasi *Android* menggunakan *Android Studio*. *Java* menyediakan konsep enkapsulasi, pewarisan, polimorfisme, serta dukungan *multithreading* yang mendukung pembangunan aplikasi yang modular dan mudah dipelihara. *Android Studio* menyediakan lingkungan pengembangan terintegrasi (IDE) yang memudahkan proses perancangan, pengkodean, pengujian, dan implementasi aplikasi *Android* (Munawaroh *et al.*, 2022). Penyimpanan data lokal seperti riwayat klasifikasi dikelola menggunakan *SQLite/Room* Database, sedangkan akses kamera menggunakan *CameraX API*. Dari sisi kompatibilitas perangkat, aplikasi ini menetapkan versi sistem operasi minimal (minimum SDK) pada *API level 24*, yaitu *Android 7.0 (Nougat)*, dan dikompilasi menggunakan *compile SDK API level 36*. Penetapan versi

OS minimal *Android 7.0 (Nougat)* dipilih agar aplikasi dapat berjalan pada sebagian besar perangkat *Android* yang masih aktif digunakan masyarakat, termasuk perangkat petani dengan spesifikasi menengah ke bawah, sekaligus tetap mendukung pustaka modern yang dibutuhkan seperti *CameraX* untuk akses kamera dan *TensorFlow Lite* untuk inferensi model CNN. Perangkat dengan versi *Android* di bawah 7.0 (*Nougat*) tidak didukung karena keterbatasan dukungan *API* yang diperlukan oleh pustaka tersebut.

Pengembangan aplikasi *Android* modern umumnya menerapkan pola arsitektur yang memisahkan logika antarmuka, logika bisnis, dan pengelolaan data agar kode lebih terstruktur dan mudah dikelola. Komponen antarmuka disusun menggunakan *Activity* dan *layout XML*, sedangkan interaksi pengguna direspons melalui *event handler*. Pemisahan tanggung jawab ini memudahkan proses pemeliharaan serta pengujian aplikasi ketika terdapat penambahan fitur baru.

Dalam aplikasi klasifikasi penyakit tanaman ini, sejumlah komponen bekerja secara terpadu, yaitu modul akuisisi citra melalui kamera atau galeri, modul inferensi yang menjalankan model *TensorFlow Lite*, serta modul penyimpanan riwayat berbasis *SQLite*. Proses inferensi yang tergolong berat dijalankan pada thread terpisah agar antarmuka tetap responsif. Dengan demikian, seluruh proses klasifikasi dapat berlangsung secara mandiri di perangkat pengguna tanpa memerlukan *server* maupun koneksi internet.

Berdasarkan uraian di atas dapat disimpulkan bahwa platform *Android* dengan bahasa *Java* menyediakan lingkungan yang tepat untuk mengimplementasikan sistem klasifikasi penyakit tanaman secara praktis. Dalam penelitian ini, aplikasi dibangun agar ringan, kompatibel dengan perangkat menengah ke bawah, dan mampu menjalankan model CNN secara *offline*.

11. *Unified Modeling Language*

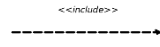
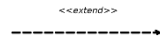
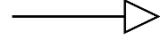
Unified Modeling Language (UML) adalah bahasa pemodelan standar yang digunakan untuk memvisualisasikan, merancang, dan mendokumentasikan sistem perangkat lunak (Wayahdi & Ruziq, 2023). UML terbagi menjadi diagram struktural dan diagram perilaku. Dalam penelitian ini digunakan tiga diagram perilaku utama: *Use Case Diagram* untuk menggambarkan fungsionalitas sistem dari sudut pandang pengguna (klasifikasi berbasis foto, melihat riwayat, dan pengelolaan data); *Activity Diagram* untuk memodelkan alur proses klasifikasi penyakit dari *input* foto daun hingga keluaran hasil; serta *Sequence Diagram* untuk menggambarkan alur komunikasi antar komponen, yaitu antarmuka pengguna (UI), model *TensorFlow Lite*, dan basis data *SQLite*.

a. **Use Case Diagram**

Use Case Diagram adalah diagram yang menggambarkan fungsionalitas sistem dari sudut pandang pengguna (aktor) serta interaksi antara aktor dengan *use case* yang tersedia. *Diagram* ini digunakan untuk mendefinisikan kebutuhan fungsional aplikasi

klasifikasi penyakit tomat dan kentang. Simbol-simbol yang digunakan pada *use case diagram* ditunjukkan pada Tabel 2.1 (Wayahdi & Ruziq, 2023).

Table 2. 1 Simbol *Use Case Diagram*

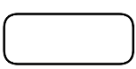
No	Simbol	Nama	Keterangan
1		<i>Actor</i>	Mewakili pengguna atau sistem eksternal yang berinteraksi dengan sistem.
2		<i>Use Case</i>	Menggambarkan fungsionalitas atau layanan yang disediakan oleh sistem.
3		<i>Association</i>	Menghubungkan <i>actor</i> dengan <i>use case</i> yang saling berinteraksi.
4		<i>System Boundary</i>	Menggambarkan batas atau ruang lingkup sistem yang dibangun.
5		<i>Include</i>	Menunjukkan suatu use case wajib memanggil (menyertakan) <i>use case</i> lain.
6		<i>Extend</i>	Menunjukkan <i>use case</i> yang dapat memperluas perilaku <i>use case</i> lain secara opsional.
7		<i>Generalization</i>	Menggambarkan hubungan pewarisan antara elemen umum dan elemen khusus.

b. Activity Diagram

Activity Diagram adalah diagram yang menggambarkan aliran kerja (*workflow*) atau urutan aktivitas dari sebuah proses dalam sistem. Pada penelitian ini *activity diagram* digunakan untuk memodelkan alur proses klasifikasi penyakit, mulai dari pengambilan/pemilihan foto daun, proses inferensi model, hingga penyajian hasil klasifikasi kepada

pengguna. Simbol-simbol yang digunakan pada *activity diagram* ditunjukkan pada Tabel 2.2 (Wayahdi & Ruziq, 2023).

Table 2. 2 Simbol *Activity Diagram*

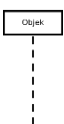
No	Simbol	Nama	Keterangan
1		<i>Initial Node</i>	Menggambarkan titik awal (status awal) dari sebuah aktivitas.
2		<i>Action/Activity</i>	Menggambarkan aktivitas atau proses yang dilakukan dalam sistem.
3		<i>Decision/Merge</i>	Menggambarkan percabangan keputusan berdasarkan kondisi tertentu.
4		<i>Fork/Join</i>	Menggambarkan aktivitas yang berjalan paralel (<i>fork</i> memecah, <i>join</i> menyatukan).
5		<i>Control Flow</i>	Menggambarkan aliran kendali dari satu aktivitas ke aktivitas berikutnya.
6		<i>Swimlane</i>	Mengelompokkan aktivitas berdasarkan aktor atau objek yang bertanggung jawab.
7		<i>Final Node</i>	Menggambarkan titik akhir (status akhir) dari sebuah aktivitas.

c. Sequence Diagram

Sequence Diagram adalah diagram yang menggambarkan interaksi antar objek dalam urutan waktu, yaitu pesan-pesan yang dikirim dan diterima antar objek selama sebuah proses berlangsung. Pada penelitian ini *sequence diagram* digunakan untuk menggambarkan alur komunikasi antar komponen, yaitu antarmuka

pengguna (UI), model *TensorFlow Lite*, dan basis data *SQLite*. Simbol-simbol yang digunakan pada *sequence diagram* ditunjukkan pada Tabel 2.3 (Wayahdi & Ruziq, 2023).

Table 2. 3 Simbol *Sequence Diagram*

No	Simbol	Nama	Keterangan
1		<i>Actor</i>	<i>Entitas</i> (pengguna) yang memulai interaksi dengan sistem.
2		<i>Object/Lifeline</i>	Mewakili objek/partisipan beserta rentang waktu hidupnya selama interaksi.
3		<i>Activation</i>	Menunjukkan periode ketika objek aktif menjalankan suatu proses.
4		<i>Message</i>	Pesan atau pemanggilan metode dari satu objek ke objek lain.
5		<i>Return Message</i>	Mengembalikan hasil atau nilai kepada objek pemanggil.
6		<i>Self Message</i>	Objek memanggil metode pada dirinya sendiri.

Berdasarkan uraian di atas dapat disimpulkan bahwa UML menyediakan notasi standar untuk memodelkan sistem secara visual sebelum tahap implementasi. Dalam penelitian ini, *Use Case Diagram*, *Activity Diagram*, dan *Sequence Diagram* digunakan untuk menggambarkan fungsionalitas, alur proses, dan interaksi antar

komponen sistem klasifikasi penyakit tanaman sehingga rancangan sistem menjadi lebih jelas dan terdokumentasi.

12. *System Usability Scale*

System Usability Scale (SUS) adalah metode pengujian *usability* (kebergunaan) berupa kuesioner standar untuk mengukur tingkat kemudahan penggunaan suatu sistem atau aplikasi dari sudut pandang pengguna. SUS terdiri dari 10 pernyataan dengan skala *Likert* 1-5 (mulai dari sangat tidak setuju hingga sangat setuju), di mana pernyataan bernomor ganjil bernada positif dan pernyataan bernomor genap bernada negatif. Skor akhir SUS dihitung dengan ketentuan: untuk pernyataan ganjil, skor = nilai jawaban - 1; untuk pernyataan genap, skor = 5 - nilai jawaban; kemudian seluruh skor dijumlahkan lalu dikalikan 2,5 sehingga menghasilkan nilai akhir pada rentang 0-100. Nilai rata-rata SUS sebesar 68 dijadikan acuan (*benchmark*): nilai di atas 68 menunjukkan *usability* di atas rata-rata, sedangkan nilai di bawah 68 menunjukkan *usability* di bawah rata-rata (Hyzy *et al.*, 2022). Dalam penelitian ini, SUS digunakan untuk mengukur tingkat kebergunaan aplikasi *Android* klasifikasi penyakit tomat dan kentang berdasarkan penilaian responden setelah mencoba seluruh fitur utama aplikasi, mulai dari pengambilan/unggah foto daun, klasifikasi CNN, tampilan hasil dan rekomendasi penanganan, hingga riwayat klasifikasi (Najaf *et al.*, 2024).

SUS banyak dipilih dalam penelitian pengembangan perangkat lunak karena bersifat ringkas, cepat diisi, serta terbukti andal (*reliable*)

meskipun jumlah responden relatif kecil. Kuesioner ini bersifat teknologi-agnostik sehingga dapat diterapkan pada berbagai jenis sistem, termasuk aplikasi *mobile*. Hasil pengukuran berupa skor tunggal pada rentang 0-100 memudahkan interpretasi dan perbandingan tingkat kebergunaan antar sistem (Hyzy *et al.*, 2022).

Selain nilai ambang 68, skor SUS juga dapat diterjemahkan ke dalam kategori peringkat (*grade*) dan tingkat penerimaan (*acceptability*), mulai dari *not acceptable*, *marginal*, hingga *acceptable*. Interpretasi ini membantu peneliti tidak hanya menilai apakah sistem dapat diterima, tetapi juga sejauh mana kualitas pengalaman penggunaannya. Dengan demikian, SUS dinilai memadai untuk mengukur keberterimaan aplikasi oleh pengguna akhir, dalam hal ini petani dan pengguna umum.

Berdasarkan uraian di atas dapat disimpulkan bahwa *System Usability Scale* merupakan instrumen baku yang efektif untuk mengukur tingkat kebergunaan aplikasi. Dalam penelitian ini, SUS digunakan sebagai metode pengujian *usability* untuk memastikan aplikasi klasifikasi penyakit tomat dan kentang mudah digunakan oleh pengguna akhir.

B. Kajian Empiris

Kajian empiris dilakukan untuk mengidentifikasi penelitian terdahulu yang relevan dengan topik klasifikasi penyakit tanaman tomat dan kentang menggunakan *Convolutional Neural Network berbasis Android*. Telah dilakukan terhadap lima penelitian terdahulu yang terdiri atas satu penelitian

internasional dan empat penelitian nasional, sebagaimana diuraikan dalam paragraf-paragraf berikut ini.

Pertama, penelitian internasional oleh Zhong *et al.*, (2023) mengembangkan *LightMixer*, yaitu model *Convolutional Neural Network* ringan untuk deteksi penyakit daun tomat. Model ini mencapai akurasi 99,3% dengan jumlah parameter yang sangat kecil (hanya 1,5 juta parameter), sehingga efisien secara komputasi dan cocok diterapkan pada perangkat *mobile*. Hubungan dengan penelitian ini terletak pada kesamaan penggunaan arsitektur CNN ringan pada citra daun tomat yang ditujukan untuk perangkat *mobile*, sedangkan perbedaannya penelitian ini menambahkan tanaman kentang, menggunakan arsitektur *MobileNetV2*, serta mengimplementasikan model secara *offline* pada perangkat *Android*.

Kedua, penelitian nasional oleh Soekarta *et al.*, (2023) melakukan klasifikasi penyakit tanaman tomat menggunakan metode CNN dan memperoleh akurasi pelatihan sebesar 98% serta akurasi validasi sebesar 82%. Penelitian tersebut menunjukkan bahwa CNN efektif untuk klasifikasi penyakit daun tomat, meskipun masih terdapat selisih (*gap*) antara akurasi pelatihan dan validasi. Hubungan dengan penelitian ini adalah kesamaan pemanfaatan CNN untuk klasifikasi penyakit daun tomat, sedangkan perbedaannya penelitian ini menggunakan arsitektur ringan *MobileNetV2* dan mengimplementasikannya berbasis *Android* sehingga dapat digunakan langsung oleh petani di lapangan.

Ketiga, penelitian nasional oleh Adhitya *et al.*, (2023) mengklasifikasikan penyakit tanaman tomat menggunakan CNN dan

mengimplementasikan model H5 pada aplikasi *desktop*. Model yang dibangun mampu mengenali beberapa jenis penyakit daun tomat dengan hasil yang cukup baik. Hubungan dengan penelitian ini adalah kesamaan dalam menerapkan model CNN ke dalam sebuah aplikasi, sedangkan perbedaannya terletak pada platform yang digunakan, yaitu *desktop* pada penelitian terdahulu dan *Android* pada penelitian ini sehingga lebih portabel dan mudah diakses pengguna.

Keempat, penelitian nasional oleh (Handayani et al., 2025) melakukan klasifikasi citra penyakit daun tomat menggunakan CNN dengan arsitektur VGG-19 dan memperoleh akurasi sebesar 97% pada data uji, dengan rata-rata presisi, *recall*, dan *F1-score* masing-masing sebesar 0,97. Penelitian tersebut menggunakan *dataset* 6.600 citra daun tomat yang terbagi dalam enam kelas penyakit yang bersumber dari *Kaggle*, dengan pendekatan *transfer learning* berbasis bobot pra-latih *ImageNet*. Hubungan dengan penelitian ini adalah kesamaan objek penelitian berupa klasifikasi penyakit daun tomat menggunakan CNN dengan *transfer learning*, sedangkan perbedaan utamanya terletak pada arsitektur yang digunakan; VGG-19 memiliki lebih dari 20 juta parameter yang dapat dilatih sehingga membutuhkan sumber daya komputasi besar dan kurang efisien untuk perangkat *mobile*, sementara penelitian ini menggunakan *MobileNetV2* yang ringan dan dioptimalkan untuk berjalan secara *offline* pada perangkat *Android*.

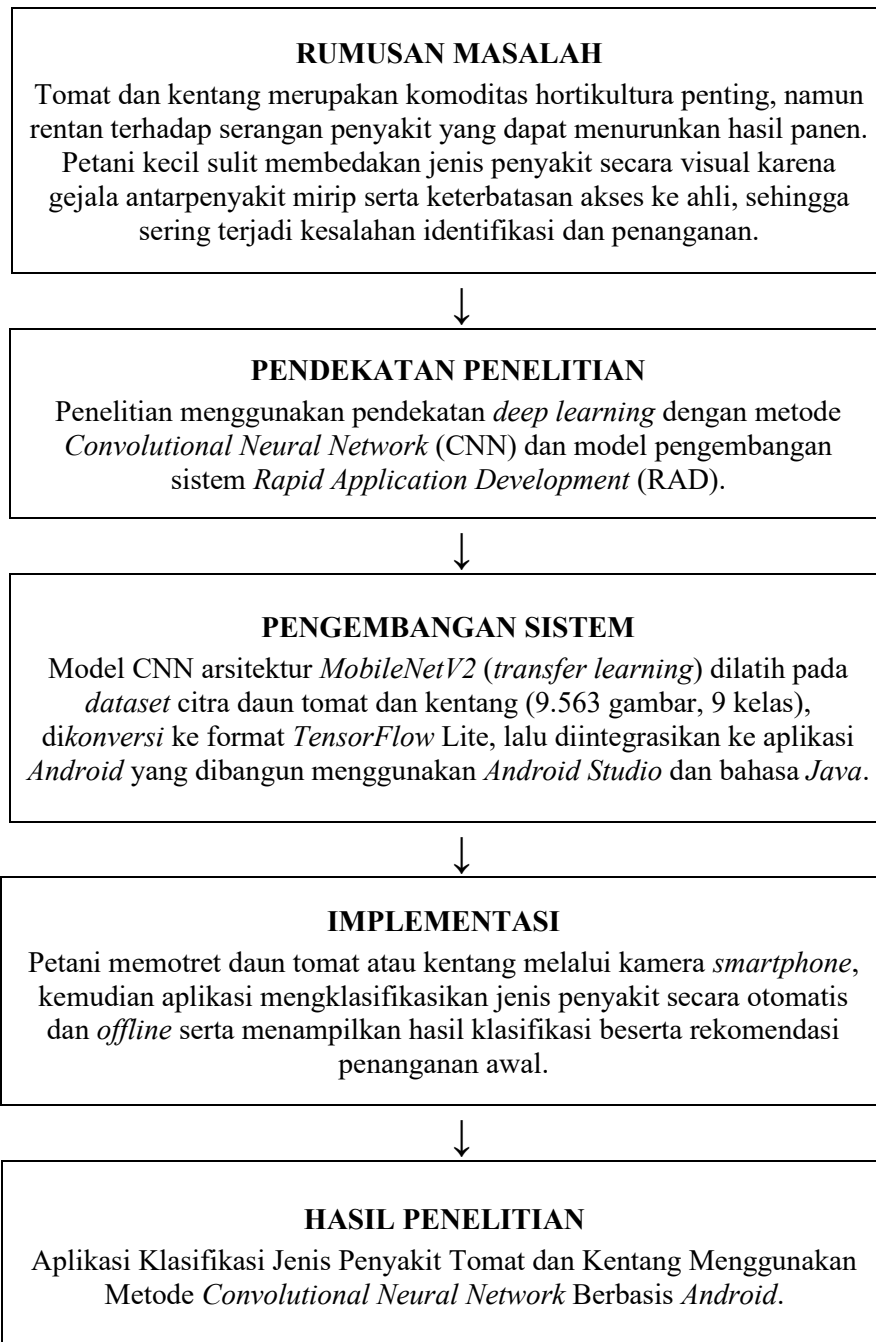
Kelima, penelitian nasional oleh Edoazani et al., (2025) mengembangkan aplikasi *Android* untuk deteksi penyakit tanaman tomat yang berbasis *cloud* dengan pola arsitektur MVVM. Hubungan dengan penelitian ini

adalah kesamaan pengembangan aplikasi pada platform *Android*, sedangkan perbedaannya penelitian terdahulu bergantung pada layanan *cloud* yang memerlukan koneksi internet, sementara penelitian ini menjalankan inferensi secara *offline* menggunakan *TensorFlow Lite* serta mencakup tanaman kentang.

Berdasarkan telaah terhadap kelima penelitian empiris di atas, penelitian ini memiliki kebaruan (*novelty*) yang terletak pada integrasi beberapa aspek utama. Pertama, pada domain tanaman, penelitian ini mencakup dua komoditas hortikultura penting di Indonesia sekaligus, yaitu tomat dan kentang, yang jarang dikombinasikan dalam satu model klasifikasi. Kedua, penelitian ini menerapkan arsitektur ringan *MobileNetV2* dengan *transfer learning* yang dikonversi ke format *TensorFlow Lite*, sehingga sistem dapat berjalan sepenuhnya secara *offline* di perangkat *Android* tanpa bergantung pada layanan *cloud* atau platform *website* seperti sebagian besar penelitian terdahulu. Terakhir, pada cakupan kelas, model ini dilatih khusus untuk mengklasifikasikan 9 kelas, yaitu 6 penyakit pada tomat dan kentang, 2 kondisi sehat, serta 1 kelas untuk objek yang tidak teridentifikasi sebagai daun tomat atau kentang, termasuk penyakit pada umbi (*Common Scab*).

C. Kerangka Berpikir

Kerangka berpikir pada penelitian ini menggambarkan alur pemikiran mulai dari rumusan masalah hingga hasil penelitian. Alur tersebut divisualisasikan dalam bentuk bagan seperti ditunjukkan pada Gambar 2.6 berikut:



Gambar 2. 6 Bagan Kerangka Berpikir