# BAB II
# LANDASAN TEORI

## 2.1 Penelitian Terdahulu

Bagian ini menjelaskan 10 penelitian terdahulu yang berkaitan dengan penelitian ini. Persamaan yang dimiliki oleh penelitian terdahulu akan dibahas bersama dengan titik kuat dan kelemahannya untuk menjelaskan *research gap.*

### 2.1.1 Pendekatan Algoritma Tradisional dan *Neural Network* Klasik

Upaya klasifikasi buah secara otomatis bergantung pada ekstraksi fitur yang dilakukan secara manual, dikombinasikan dengan algoritma *machine learning* tradisional. Tabel 2.1 menunjukan sebuah studi mengenai penyakit daun kiwi menggunakan CNN-SVM dengan rasio pelatihan dan pengujian yang sama serta jumlah klasifikasi penyakit yang sama, dengan akurasi sebesar 83,34 [11]. Dalam kasus serupa, [12] menggunakan *neural network multilayer perceptron* yang dilatih pada fitur warna dan tekstur untuk klasifikasi penyakit apel. Meskipun pendekatan *neural network* ini menunjukkan kontribusi penting dan memberikan dataset baru bagi bidang ini, akurasinya relatif rendah (73,3%). Hal ini menunjukkan batasan mendasar di mana fitur pembelajaran mesin khusus tidak mampu menangkap pola kompleks yang membedakan berbagai jenis apel. Metode-metode ini memerlukan modifikasi fitur yang melewatkan indikator visual kritis. Meskipun demikian, penelitian sebelumnya menunjukkan diagnosis otomatis yang tepat dan menyediakan dasar bagi pendekatan penelitian serupa.

Tabel 2.1 Studi Literatur Pendekatan Algoritma Tradisional dan Neural Network Klasik

| Artikel | Metode | Performa | Ringkasan |
|---|---|---|---|
| [11] | CNN-SVM | Akurasi 83,4% | Penelitian ini berfokus pada klasifikasi penyakit pada daun kiwi, yang diklasifikasikan ke dalam 5 kategori. |

| | | | |
|---|---|---|---|
| | | | Antraknosa merupakan salah satu kategori tersebut. Penyakit ini sama dengan yang digunakan dalam salah satu penelitian terkini. Akurasi penelitian ini masih dapat ditingkatkan. Hal ini menunjukkan bahwa model pembelajaran mesin belum mampu membedakan berbagai penyakit yang tampak serupa dengan baik. |
| [12] | ANN-MLP | 73,3% | Penelitian ini memiliki kesamaan dalam hal klasifikasi penyakit apel berbasis citra serta menggunakan dataset yang relevan. Namun, pendekatannya masih menggunakan arsitektur ANN sederhana yang kemampuannya terbatas dalam mengekstrak fitur kompleks pada citra. |

### 2.1.2 Pendekatan *Lightweight* Berbasis *Channel attention*

Untuk mengatasi beban komputasi yang besar pada perangkat dengan sumber daya terbatas, tren riset beralih ke arsitektur backbone ringan yang dioptimalkan menggunakan mekanisme *channel attention* seperti yang dapat dilihat pada Tabel *2.2*. Studi [13] memvalidasi efisiensi MobileNetV2 untuk lingkungan seluler dengan akurasi 93,70%, di mana mekanisme *Efficient Channel attention* (ECA) diterapkan untuk menyeimbangkan akurasi dan ukuran model (22,3 MB). Selanjutnya, [14] berfokus pada inefisiensi modul bawaan MobileNetV3 dengan menggantinya menggunakan konvolusi 1D melalui ISNECA, menghasilkan model efisien berakurasi 91%. Optimasi interaksi antar-saluran ini diperluas oleh [3], yang mengintegrasikan ECA-Net ke dalam MobileNetV2 dengan koneksi padat (dense connections). Pendekatan ini berhasil mencapai akurasi tinggi sebesar 96,3% dengan parameter yang tetap minimal (3,3 M), meskipun penggunaan GAN pada pemrosesan datanya

meningkatkan kompleksitas *pipeline*. Namun, metode ini mengabaikan informasi spasial seperti lokasi, bentuk, dan persebaran gejala bercak pada buah apel sehingga akurasinya cenderung stagnan.

Tabel 2.2 Pendekatan *Lightweight* Berbasis *Channel attention*

| Artikel | Metode | Performa | Ringkasan |
|---|---|---|---|
| [13] | MobileNetv2 | Akurasi 93.70%, jumlah parameter 2.13, ukuran 22.3 MB | Penelitian ini memiliki kesamaan berupa penggunaan arsitektur MobileNet sebagai *backbone* serta penerapan mekanisme *attention* (ECA). Namun, mekanisme ECA yang digunakan hanya diterapkan pada dua lapisan bottleneck tertentu dan belum mengintegrasikan *attention* spasial. Ukurannya pun masih bisa dikembangkan agar menjadi lebih ringan. |
| [14] | ISNECA-Mobilenetv3 | Akurasi 91% | Penelitian ini memiliki kesamaan berupa penggunaan MobileNetV3 serta *attention* yang berfokus terhadap efisiensi komputasi. Namun, penelitian ini belum mempelajari potensi *hybrid* dengan *spatial attention*. Akurasi juga masih dapat ditingkatkan yang menandakan pendekatan ECA belum dapat melakukan deteksi pada domain yang lebih spesifik. |
| [3] | ECANet-MobileNetv2 | Akurasi 96,3%, parameter 3.3 M | Penelitian ini memiliki kesamaan berupa klasifikasi penyakit apel berbasis citra dan penggunaan MobileNet untuk *backbone* ringan. Tidak hanya itu, model ini juga menggunakan penggabungan dengan *attention* serta fokus terhadap model efisien dan ringan untuk deployment. |

| | | | |
|---|---|---|---|
| | | | Penggunaan GAN untuk augmentasi data menambah kompleksitas *pipeline* peltihan serta belum mengoptimalkan *attention* untuk informasi spasial lokasi dan persebaran gejala. |

### 2.1.3 Pendekatan Mekanisme *Spatial attention* dan Multidimensi

*Channel attention* sering kali mengabaikan informasi posisi visual gejala penyakit yang tersebar. Oleh karena itu, Tabel 2.3 berfokus pada integrasi elemen spasial atau struktural (multidimensi) untuk penyempurnaan fitur yang lebih terlokalisasi. Sebuah studi mengembangkan GrapeNet yang mengombinasikan blok residual dengan modul CBAM (yang menggabungkan dimensi saluran dan spasial), menghasilkan akurasi 89% namun mengalami stagnasi akibat kerumitan arsitektur [9]. Di sisi lain, studi lain mengusulkan modul *Spatial-ECA Hybrid* pada kerangka ResNet50 yang menghubungkan *spatial attention* dan saluran secara seri, terbukti meningkatkan stabilitas pengenalan penyakit tanaman meskipun peningkatan akurasinya dari *base*line relatif kecil (0,78%) [8]. Sementara itu, sebuahs studi lain memperkenalkan *Spatial Temporal Awareness Module* (STAM) melalui arsitektur DeepMAD untuk memproses cabang fitur struktural secara paralel [15]. Walaupun akurasinya berada di angka 89%, penggunaan arsitektur khusus ini mampu memangkas parameter secara ekstrem hingga tersisa 401.000 parameter saja.

Tabel 2.3 Pendekatan Mekanisme *Spatial attention* dan Multidimensi

| Artikel | Metode | Performa | Ringkasan |
|---|---|---|---|
| [9] | Grapenet | Akurasi Rata-rata 89%, jumlah parameter 2.15 M | Penelitian terhadap penyakit buah anggur ini memiliki kesamaan dalam penggunaan blok residual dan modul perhatian untuk menciptakan model yang lebih ringkas daripada model dasar. Namun, hal ini mengorbankan akurasi demi keunggulan dalam hal kerampingan. |

| | | | |
|---|---|---|---|
| [8] | ECA-*Spatial Attention Hybrid* ResNet50 | Akurasi 87% | Penelitian ini memiliki kesamaan berupa penggabungan *channel attention* dan *spatial attention* untuk meningkatkan fokus model pada fitur penyakit. Namun, akurasi masih perlu ditingkatkan dikarenakan hanya mendapat peningkatan akurasi sebanyak 0.78% dari model *base*line. |
| [15] | DeepMAD with STAM Module | Accuracy 89%, 401k Parameters | Penelitian ini memiliki kesaman berupa penggunaan *channel* dan *spatial attention*. Namun, akurasi masih dapat ditingkatkan dan memiliki desain arsitektur yang kompleks. *Backbone* yang diperoleh melalui NAS juga memerlukan proses search arsitektur tambahan sehingga menambahkan komplesitas *pipeline*. |

### 2.1.4 Pendekatan Kompresi Arsitektur Hibrida dan Implementasi Perangkat Seluler

Tabel 2.4 menunjukan pendekatan pada literatur terkini yang menggabungkan kerangka kerja ringan dengan mekanisme *attention* dan teknik pembangkitan fitur yang efisien. Sebuah studi meningkatkan MobileNetv3 dengan modul *Ghost*, yang menghasilkan peta fitur redundan melalui operasi linier yang murah daripada konvolusi yang mahal [7]. Dikombinasikan dengan *attention* CBAM, arsitektur ini mencapai akurasi 98% dengan 2.08M yang menunjukkan keseimbangan yang cukup baik antara kinerja dan efisiensi yang masih dapat ditingkatkan. Namun, penggunaan CBAM standar berarti cabang *attention* spasial masih bergantung pada operasi *channel* pooling yang membuang hubungan antar-saluran yang berpotensi berguna. Studi lain mengusulkan jaringan MobileNetV3 dual-stream yang menggabungkan blok residual terbalik, koneksi lintas lapisan, dan modul ECA [16]. Integrasi pembelajaran transfer dan penerapan pada Android dengan akurasi 98,77%

pada klasifikasi penyakit tomat menunjukkan kegunaan praktis pendekatan hibrida semacam itu untuk aplikasi pertanian di dunia nyata. Namun, arsitektur *dual-stream* meningkatkan kebutuhan komputasi dibandingkan desain *single-stream*, dengan kemungkinan membatasi penerapan pada perangkat yang terbatas (*resource constrained device*).

Tabel 2.4 Pendekatan Kompresi Arsitektur Hibrida dan Implementasi Perangkat Seluler

| Artikel | Metode | Performa | Ringkasan |
|---|---|---|---|
| [7] | MobileNetv3 dengan *Ghost Module-CBAM hybrid* | Akurasi 98,63%, jumlah parameter 2.08M | Penelitian ini memiliki kesamaan berupa penggunaan mekanisme *attention* dan MobileNetV3 sebagai *backbone* ringan serta menerapkan metode *transfer learning*. Namun, penggunaan Ghost Module dan CBAM dapat menambah kompleksitas model. |
| [16] | Modifikasi CNN *Module* pada MobileNetv3-ECA *Hybrid* | Akurasi 98,77% | Penelitian ini memiliki kesamaan berupa pengunaan MobileNetV3 sebagai *backbone* yang ringat serta penggunaan *attention*. Tidak hanya itu, model ini menerpkan *transfer learning* dengan penyesuaian khusus. Namun, arsitektur *dual-stream* ini dapat mempersulit proses training karena kompleksitas desain. |

### *2.1.4 Research Gap*

Penelitian terdahulu mengenai klasifikasi penyakit tanaman umumnya masih menghadapi dilema antara akurasi dan efisiensi komputasi. Model ringan konvensional yang mengandalkan *channel attention* murni (seperti MobileNet standar) terbukti hemat dalam aspek parameter, namun gagal menangkap informasi spasial seperti bentuk dan lokasi gejala penyakit pada buah apel,

sehingga akurasinya cenderung stagnan. Sebaliknya, upaya mengintegrasikan fitur spasial melalui modul multidimensi standar seperti CBAM, atau penggunaan arsitektur hibrida *dual-stream*, justru meningkatkan kompleksitas arsitektur dan konsumsi memori (RAM). Akibatnya, model-model tersebut menjadi berat dan tidak praktis untuk diterapkan pada perangkat dengan sumber daya terbatas (resource-constrained devices)

*Reserach gap* inilah yang menjadi fokus dalam penelitian tesis ini, di mana belum ada studi yang mengeksplorasi kombinasi antara arsitektur truncated MobileNetV3 dengan mekanisme *Lightweight Sequential Channel and Spatial Attention* (LiteSCSA) untuk klasifikasi penyakit buah apel. Solusi yang diajukan menggunakan dua strategi : pemotongan lapisan (*truncation*) pada MobileNetV3 dilakukan untuk memangkas redundansi parameter secara ekstrem, sementara modul atensi sekuensial yang ringan diintegrasikan untuk mempertajam akurasi fitur spasial tanpa melibatkan operasi *pooling* yang berat. Melalui pendekatan hibrida terkompresi ini, dihasilkan model klasifikasi penyakit apel yang tidak hanya presisi, tetapi efisien untuk diimplementasikan secara optimal pada perangkat edge yang terbatas.

## 2.2 Bisnis / Topik

### 2.2.1 Klasifikasi Penyakit Buah Apel

Klasifikasi penyakit buah apel merupakan proses identifikasi jenis penyakit yang menyerang tanaman apel berdasarkan karakteristik visual yang tampak pada permukaan buah. Penyakit buah apel menjadi salah satu faktor utama penurunan produktivitas tanaman apel di berbagai sentra produksi. Deteksi dini melalui citra digital memungkinkan petani untuk melakukan tindakan pengendalian secara cepat dan tepat. Dalam konteks pengolahan citra digital, fitur warna, tekstur, dan pola bercak pada buah menjadi parameter kunci untuk membedakan jenis penyakit. Penelitian ini berfokus pada lima kelas kondisi buah apel, yaitu:

a) *Healthy*

Buah apel yang sehat menjadi referensi utama dalam sistem klasifikasi. Kondisi buah sehat digunakan sebagai acuan pembanding untuk mendeteksi anomali visual yang mengindikasikan adanya infeksi penyakit seperti pada Gambar 2.1. Sistem diagnosis otomatis yang efisien dapat secara cepat mendiagnosis dan mengendalikan penyakit sehingga *loss* dapat diminimalkan melalui deteksi sampel, klasifikasi menggunakan metode pengolahan citra, dan ekstraksi fitur seperti warna dan tekstur.

*Gambar 2.1 Buah Apel Sehat [17]*

*b) Black Rot*

Buah apel yang sehat menjadi referensi utama dalam sistem klasifikasi. Gambar 2.2 menunjukkan kondisi buah sehat digunakan sebagai acuan pembanding untuk mendeteksi anomali visual yang mengindikasikan adanya infeksi penyakit. Sistem diagnosis otomatis yang efisien dapat secara cepat mendiagnosis dan mengendalikan penyakit sehingga *loss* dapat diminimalkan melalui deteksi sampel, klasifikasi menggunakan metode pengolahan citra, dan ekstraksi fitur seperti warna dan tekstur.

Gambar 2.2 Buah Apel dengan Penyakit *Black Rot [17]*

c) *Powdery Mildew*

Penyebabnya adalah cendawan *Podosphaera leucotricha* [18]. Gejalanya adalah lapisan miselium dan konidia berwarna putih keabuan seperti tepung yang menutupi permukaan buah, umumnya di sisi bawah yang dapat dilihat pada Gambar 2.3.

Gambar 2.3 Buah Apel dengan Penyakit *Powdery Mildew[17]*

d) *Black Pox*

*Black Pox* pada apel menghasilkan titik-titik *pycnidia* hitam (struktur penghasil spora dari jamur) yang muncul pada permukaan atas bercak buah yang lebih tua, serta bercak kecil melingkar berwarna cokelat yang dikelilingi cincin konsentris berwarna cokelat gelap [19]. Ciri khas ini membantu membedakan *Black Pox* dari bercak-bercak serupa yang disebabkan oleh kerusakan semprot Gambar 2.4.

Gambar 2.4 Buah Apel dengan Penyakit *Black Pox[17]*

e) *Anthracnose*

Penyakit antraknosa pada apel disebabkan oleh *Colletotrichum gloeosporioides* (fase teleomorf *Glomerella cingulata*). Pada buah, timbul bercak cokelat tidak beraturan dengan tepi gelap [18]. Gambar 2.5 menunjukan bercak dapat mengering dan menyebabkan bagian tengah rontok sehingga membentuk lubang (*shot hole*). Dalam kondisi lembap, kadang terlihat titik-titik hitam (*aservulus*) yang merupakan tubuh buah cendawan.

Gambar 2.5 Buah Apel dengan Penyakit *Anthracnose [17]*

### 2.2.2 *Channel attention*

*Attention mechanism* dalam *deep learning* terinspirasi dari sistem persepsi visual manusia, di mana otak secara selektif memusatkan perhatian pada bagian informasi yang penting dan mengabaikan informasi lain yang kurang relevan. Keunggulan utama dari *channel attention* adalah kemampuannya untuk meningkatkan akurasi klasifikasi dan membuat model lebih kebal (*robust)* terhadap variasi latar belakang atau derau, dengan penambahan parameter dan beban komputasi (*overhead)* yang minimal. Oleh karena itu, mekanisme ini ideal untuk diintegrasikan pada arsitektur model ringan yang ditujukan untuk perangkat dengan sumber daya terbatas [20].

Cara kerja mekanisme *channel attention* dimulai dengan mengevaluasi tingkat kepentingan setiap saluran (*channel*) pada peta fitur melalui tahap

kompresi (*squeeze)*, di mana dimensi spasial (lebar dan tinggi) direduksi menjadi satu nilai statistik tunggal menggunakan operasi *Global Pooling*. Selanjutnya, pada tahap eksitasi (*excitation*), nilai-nilai yang telah dikompresi tersebut dilewatkan ke dalam sebuah neural network sederhana yang diakhiri dengan fungsi aktivasi Sigmoid guna mempelajari interaksi antar-saluran dan menghasilkan bobot probabilitas bernilai antara 0 hingga 1. Pada tahap akhir, matriks bobot yang telah dipelajari ini dikalikan kembali secara elemen-demi-elemen (*element-wise*) dengan peta fitur aslinya, sehingga secara dinamis model akan memperkuat saluran yang berisi representasi fitur-fitur penting (seperti pola warna atau tekstur penyakit) dan menekan (*suppress*) saluran yang hanya memuat derau atau informasi latar belakang yang tidak relevan.

### 2.2.3 *Spatial Attention*

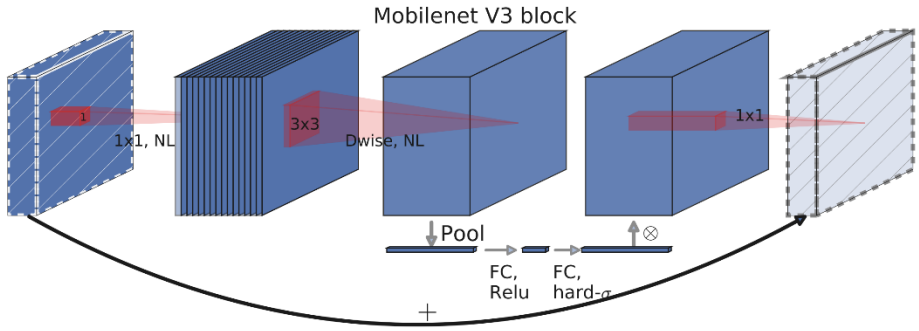
Mekanisme *spatial attention* dirancang untuk menjawab kordinat lokasi informasi yang paling diskriminatif tersebut berada pada sebuah citra. Dalam konteks *computer vision*, tidak semua area dalam sebuah citra memiliki nilai informasi yang sama. Sebagian besar area mungkin hanya berupa latar belakang (*background*) yang tidak relevan, seperti daun sehat, ranting, atau tanah pada citra pertanian.

*Spatial attention* bekerja dengan cara mengekstraksi hubungan spasial antar-piksel dalam *feature map*. Tujuan utamanya adalah untuk membuat *spatial* mask yang akan fokus memerhatikan area yang berisi objek target utama dan meredam aktivasi pada area latar belakang yang dapat menjadi derau *noise* bagi model.

### 2.2.4 MobileNetv3

MobileNetV3 merupakan arsitektur *Convolutional neural network* (CNN) generasi terbaru yang dirancang khusus oleh Google untuk memberikan efisiensi komputasi maksimal pada perangkat seluler dan sistem tertanam (embedded systems). Model ini dikembangkan dengan memanfaatkan kombinasi teknik pencarian arsitektur otomatis, yaitu *Neural Architecture*

*Search* (NAS) dan algoritme *NetAdapt*, untuk menemukan konfigurasi lapisan yang paling optimal dalam menyeimbangkan antara akurasi, latensi, dan penggunaan daya [17]. MobileNetV3 tetap mempertahankan keunggulan struktur *Inverted Residual* dan *Depthwise Separable Convolution* dari versi sebelumnya, namun dengan penyempurnaan pada desain blok penyusunnya untuk memangkas redundansi parameter tanpa menurunkan performa ekstraksi fitur secara signifikan seperti pada Gambar 2.6.




Gambar 2.6 Arsitektur MobileNetV3 *[21]*

Dalam penelitian ini, digunakan varian MobileNetV3-Small dengan faktor yang telah dilatih sebelumnya (*pretrained*) pada dataset ImageNet. Pemilihan varian ini didasarkan pada keseimbangan antara akurasi dan efisiensi komputasi yang sesuai dengan kebutuhan klasifikasi penyakit buah apel.

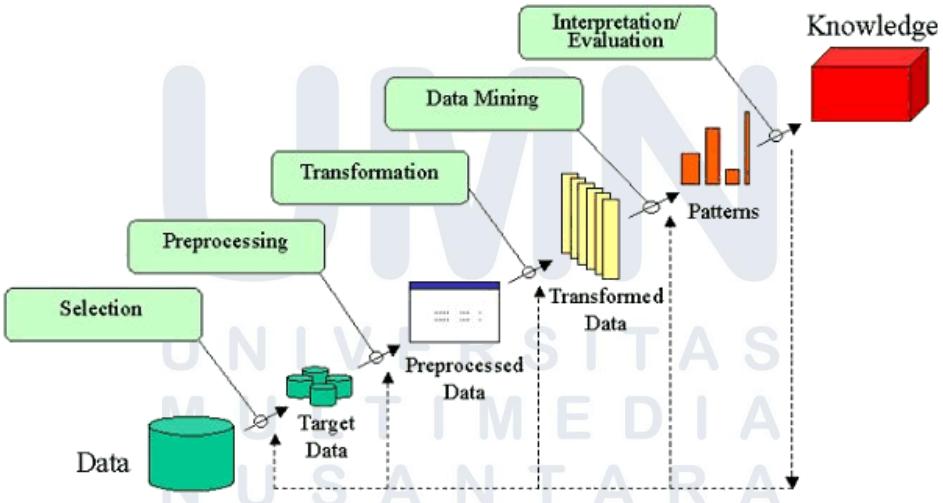
### 2.2.5 *Transfer learning*

Dalam pendekatan machine learning tradisional, sebuah model dibangun dan dilatih dari awal untuk menyelesaikan satu tugas spesifik. Proses ini membutuhkan kumpulan data latih berskala masif, waktu pelatihan yang lama, serta daya komputasi yang besar. *Transfer learning* hadir sebagai paradigma yang memecahkan masalah tersebut dengan cara mentransfer "pengetahuan" yang telah dipelajari oleh sebuah model dari satu tugas awal ke tugas baru yang saling berkaitan (*target task)* [22].

Secara konseptual, model *deep learning* (seperti CNN) yang dilatih pada dataset yang besar seperti ImageNet yang berisi jutaan citra dari ribuan kelas. akan secara otomatis mempelajari representasi fitur visual yang kaya dan hierarkis. Pada lapisan-lapisan awal (*early layers*), model mempelajari fitur-fitur generik seperti garis, tepi (*edges*), kurva, dan bercak warna. Representasi fitur generik inilah yang berharga dan dapat digunakan ulang (*reused)* untuk mengenali objek pada domain data yang sama sekali baru.

### 2.2.6 KDD (*Knowledge Discovery in Databases*)

*Knowledge Discovery in Databases (KDD)* adalah proses terorganisir untuk mengekstraksi informasi berharga, pola, atau pengetahuan yang sebelumnya tidak diketahui dari kumpulan data yang besar dan kompleks. Metodologi ini bersifat iteratif dan sekuensial, di mana hasil dari satu tahap dapat mengharuskan peneliti kembali ke tahap sebelumnya untuk melakukan penyempurnaan guna memastikan kualitas pengetahuan yang dihasilkan baik seperti yang dapat dilihat pada Gambar 2.7 [13].




Gambar 2.7 Diagram KDD[23]

*a) Data Selection*

Tahap awal ini berfokus pada identifikasi dan pemilihan data yang relevan dari berbagai sumber data yang tersedia untuk tujuan penelitian. Dalam konteks

klasifikasi citra, hal ini melibatkan penentuan dataset citra yang akan digunakan untuk melatih model.

*b) Pre-Processing*

Langkah ini bertujuan untuk meningkatkan kualitas data dengan melakukan pembersihan (*cleaning*). Prosesnya mencakup penanganan data yang hilang (*missing values*), pengurangan gangguan (*noise*), serta penanganan data yang tidak konsisten agar siap untuk diproses lebih lanjut.

*c) Transformation*

Data yang telah dibersihkan diubah ke dalam format atau representasi yang sesuai untuk algoritma pembelajaran mesin. Kegiatannya meliputi normalisasi, reduksi dimensi, hingga teknik augmentasi data untuk memperkaya variasi informasi dalam dataset.

*d) Data Mining*

Ini merupakan tahap inti di mana algoritma cerdas, seperti *Deep learning* atau *Convolutional neural network*, diterapkan untuk mengekstraksi pola-pola dari data yang telah ditransformasi. Proses ini melibatkan pelatihan model dan penyetelan hyperparameter untuk menemukan pola visual yang membedakan antar kelas.

*e) Evaluation*

Tahap evaluasi bertujuan untuk menerjemahkan pola yang dihasilkan model menjadi pengetahuan yang mudah dipahami. Hasil evaluasi menggunakan metrik seperti akurasi dan *confusion matrix* digunakan untuk memutuskan apakah pengetahuan yang ditemukan valid dan memenuhi tujuan penelitian yang telah ditetapkan.

*f) Knowledge Discovery*

Pengetahuan yang ditemukan bukan sekadar angka atau prediksi, melainkan pemahaman mendalam mengenai karakteristik unik dari data yang diteliti. Dalam klasifikasi citra, hal ini mencakup kemampuan model untuk

mengidentifikasi fitur visual yang krusial dalam membedakan satu objek dengan objek lainnya, sehingga informasi tersebut dapat digunakan untuk pengambilan keputusan atau pengembangan sistem yang lebih cerdas di masa depan.

### 2.2.7 Metrik Akurasi

Rumus 2.1 merupakan metrik evaluasi yang paling intuitif dalam klasifikasi citra. Metrik ini mengukur sejauh mana model mampu memprediksi kelas target secara benar dengan menghitung rasio antara jumlah prediksi yang tepat (baik positif maupun negatif) terhadap total keseluruhan data yang diuji. Secara matematis, akurasi dirumuskan sebagai berikut:

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

Rumus 2.1 Akurasi

- TP (*True Positive*) = Jumlah data yang benar-benar termasuk kelas positif dan berhasil diprediksi positif oleh model.
- TN (*True Negative*) = Jumlah data yang benar-benar termasuk kelas negatif dan berhasil diprediksi negatif oleh model.
- FP (*False Positive*) = Jumlah data yang sebenarnya termasuk kelas negatif, tetapi salah diprediksi sebagai positif oleh model.
- FN (*False Negative*) = Jumlah data yang sebenarnya termasuk kelas positif, tetapi salah diprediksi sebagai negatif oleh model.

Meskipun akurasi memberikan citraan umum mengenai performa model, metrik ini dapat memberikan interpretasi yang bias jika dataset yang digunakan memiliki distribusi kelas yang tidak seimbang. Oleh karena itu, penggunaan akurasi dalam penelitian ini perlu didampingi oleh metrik lainnya untuk memastikan objektivitas hasil.

### 2.2.8 Metrik *Precision*

*Precision* atau nilai prediksi positif mengukur tingkat ketepatan model dalam mengidentifikasi kelas positif. Fokus utama dari metrik ini adalah meminimalkan kesalahan False Positive (FP). Rumus dari *precision* adalah:

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

Rumus 2.2 Rumus *Precision*

- TP (*True Positive*) = Jumlah data yang benar-benar termasuk kelas positif dan berhasil diprediksi positif oleh model.
- FP (*False Positive*) = Jumlah data yang sebenarnya termasuk kelas negatif, tetapi salah diprediksi sebagai positif oleh model.

Nilai *precision* yang tinggi menunjukkan bahwa model memiliki kemampuan yang baik dalam memberikan label kelas tanpa banyak melakukan kesalahan klasifikasi pada objek yang salah.

### 2.2.9 Metrik *Recall*

*Recall*, yang juga dikenal sebagai *Sensitivity* atau *True Positive Rate*, mengukur kemampuan model dalam menemukan kembali (*retrieve*) seluruh sampel yang termasuk dalam kelas positif yang sebenarnya. Metrik ini menjawab pertanyaan: "Dari seluruh buah yang sebenarnya matang di dalam dataset, berapa banyak yang berhasil dideteksi dengan benar oleh model?". *Recall* krusial untuk menghindari *False Negative* (FN), dan dihitung dengan rumus:

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

Rumus 2.3 Rumus *Recall*

- TP (*True Positive*) = Jumlah data yang benar-benar termasuk kelas positif dan berhasil diprediksi positif oleh model.
- FN (*False Negative*) = Jumlah data yang sebenarnya termasuk kelas positif, tetapi salah diprediksi sebagai negatif oleh model.

### 2.2.10 Metrik F1-Score

F1-Score merupakan rata-rata harmonik antara *precision* dan *recall*. Metrik ini digunakan untuk mendapatkan nilai tunggal yang mampu merepresentasikan keseimbangan antara ketepatan (*precision*) dan sensitivitas (*recall*). Penggunaan rata-rata harmonik dipilih karena metrik ini memberikan penalti yang signifikan jika salah satu dari *precision* atau *recall* memiliki nilai yang rendah. Rumusnya adalah sebagai berikut:

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

*Rumus 2.4 Rumus F1-Score*

Metrik ini berguna untuk mengevaluasi model pada dataset klasifikasi penyakit buah apel yang mungkin memiliki sebaran data yang tidak merata di setiap fasenya.

### 2.2.11 Metrik GFLOPS (*Giga Floating Point Operations Per Second*)

GFLOPS adalah metrik yang digunakan untuk mengukur kompleksitas komputasi suatu model *deep learning*. Metrik ini menghitung jumlah total operasi matematika yang dilakukan oleh model selama proses *forward pass* atau inferensi satu citra.

Dalam pengembangan model *lightweight*, GFLOPS menjadi parameter krusial untuk menentukan apakah model tersebut cukup ringan untuk dijalankan pada perangkat dengan daya komputasi rendah (*edge devices*) seperti perangkat Android. Semakin kecil nilai GFLOPS, semakin cepat waktu eksekusi model dan semakin rendah konsumsi daya baterai pada perangkat seluler. Rumusnya adalah sebagai berikut :

$$k = \psi(C) = \left| \frac{\log_2(C)}{\gamma} + \frac{b}{\gamma} \right|_{odd}$$

Rumus 2.5 Rumus GFlOPs [24]

- $C$ = Jumlah *channel* masukan pada lapisan konvolusi yang sedang di proses.

- γ= Hiperparameter pengontrol laju pertumbuhan kernel terhadap C.
- $b$= Bias (offset) yang menggeser kurva logaritmik. Defaultnya adalah 1
- $\log_2(C)$ = Logaritma basis 2 dari C. Menangkap hubungan non-linier antara jumlah *channel* dan area cakupan interaksi antar-*channel*.

### 2.2.12 Metrik t-SNE *Graph (t-Ditributed Stochastic Neighbor Embedding*)

Metrik ini menggunakan teknik reduksi dimensionalitas non-linear yang dikembangkan oleh *Laurens van der Maaten* dan *Geoffrey Hinton* pada tahun 2008. Teknik ini dirancang khusus untuk memvisualisasikan data berdimensi tinggi ke dalam ruang berdimensi rendah (biasanya 2D atau 3D) dengan mempertahankan struktur kesamaan lokal dari data asli [25].

Berbeda dengan teknik reduksi linear seperti *Principal Component Analysis* (PCA) yang mempertahankan varians global, t-SNE berfokus pada pemeliharaan struktur tetangga lokal (*local neighborhood structure*) dari data. Prinsip kerjanya didasarkan pada konversi kesamaan antar titik data menjadi probabilitas bersyarat dan kemudian meminimalkan divergensi antara distribusi probabilitas di ruang berdimensi tinggi dan ruang berdimensi rendah.

### 2.2.13 Metrik Grad-CAM (*Gradient-weighted Class Activation Mapping*)

Grad-CAM menggunakan informasi gradien yang mengalir ke lapisan konvolusional terakhir CNN untuk memahami setiap neuron bagi keputusan yang diminati [26]. Satu kelemahan CAM adalah ia membutuhkan feature maps tepat sebelum lapisan softmax, sehingga hanya berlaku pada arsitektur CNN tertentu yang melakukan global average pooling atas peta konvolusional secara langsung sebelum prediksi.

Secara lebih rinci, alur kerja Grad-CAM terdiri dari empat tahap utama:

a) Perhitungan Gradien: Gradien dari skor kelas target $y^c$dihitung terhadap *feature maps* $A^k$dari lapisan konvolusional yang dipilih, yaitu $\frac{\partial y^c}{\partial A^k}$. Gradien-

gradien ini menunjukkan seberapa penting setiap feature map terhadap kelas target.

b) *Global Average Pooling Gradien*: Setiap filter dirangkum menjadi satu angka kepentingan dengan cara merata-ratakan seluruh nilai gradien yang tersebar di peta aktivasinya, sehingga diketahui seberapa besar peran filter tersebut dalam memprediksi kelas target.

c) Kombinasi Linear dengan ReLU: Untuk menghitung peta lokalisasi Grad-CAM akhir, setiap feature map $A^k$dikalikan dengan bobot kepentingannya $\alpha_k^c$dan kemudian dijumlahkan di seluruh filter. Aktivasi ReLU diterapkan untuk hanya mempertahankan bagian yang berpengaruh positif terhadap kelas target:

$$L^c_{Grad\text{-}CAM} = \text{ReLU}\left(\sum_k \alpha_k^c \, A^k\right)$$

- $\alpha_k^c$: Skalar yang mengukur seberapa penting filter k untuk kelas
- cA$^k$: Matriks aktivasi penuh dari filter-k
- Σk: Menggabungkan semua feature map berbobot dari tiap filter
- $L^c_{Grad\text{-}CAM}$: Peta lokalisasi kasar yang lalu di-upsample ke ukuran citra asli

d) Upsampling dan Visualisasi Heatmap**:** Peta Grad-CAM yang dihasilkan kemudian di-*upsample* untuk mencocokkan dimensi citra input dan ditumpangkan (*overlay*) pada citra input agar hasilnya dapat diinterpretasikan secara visual. Wilayah dengan intensitas warna tinggi (umumnya merah atau kuning pada skema warna *jet*) merepresentasikan area yang paling berpengaruh terhadap keputusan klasifikasi model.

## 2.3 Tools/*software* yang digunakan

### 2.3.1 Python

Python adalah bahasa pemrograman tingkat tinggi yang serbaguna,

dinamis, dan mudah dipahami, yang dirancang untuk meningkatkan produktivitas pengembang dengan sintaks yang sederhana dan jelas.

### 2.3.2 Google Colab

*Google Colab (Colaboratory)* adalah platform berbasis cloud yang disediakan oleh Google untuk menulis dan menjalankan kode Python langsung dari browser. Ini adalah varian dari *Jupyter Notebook* yang telah diintegrasikan dengan *Google Drive*, sehingga penyimpanan file menjadi lebih mudah. Tidak hanya itu, *platform* ini menawarkan akses ke sumber daya komputasi bertenaga tinggi, seperti unit pemrosesan grafis atau GPU, yang memungkinkan proses komputasi berat berjalan jauh lebih efisien dibandingkan menggunakan perangkat keras lokal. Berbagai pustaka pemrograman Python yang populer, seperti Pandas, Numpy, dan TensorFlow, telah terpasang secara bawaan sehingga pengguna dapat langsung fokus pada penulisan kode melalui peramban web.

### 2.3.3 Capacitor

Capacitor merupakan runtime lintas platform sumber terbuka yang dikembangkan oleh tim Ionic. Teknologi ini memungkinkan pengembang untuk membangun aplikasi mobile native untuk iOS dan Android, serta aplikasi web progresif, menggunakan teknologi web standar seperti HTML, CSS, dan JavaScript [27] . Dalam arsitektur pengembangan perangkat lunak, Capacitor berfungsi sebagai jembatan yang menghubungkan basis kode web dengan antarmuka pemrograman aplikasi (API) bawaan dari sistem operasi seluler yang dituju. Penggunaan Capacitor memberikan tingkat efisiensi yang tinggi karena memungkinkan pemanfaatan satu basis kode utama untuk didistribusikan ke berbagai platform, sekaligus tetap memberikan akses penuh ke fitur-fitur perangkat keras ponsel seperti kamera, sistem berkas, dan sensor perangkat.

### 2.3.4 Next.js

Next.js adalah kerangka kerja pengembangan web berbasis React yang diciptakan oleh *Vercel* untuk membangun aplikasi *fullstack*. Kerangka kerja ini berfokus pada produksi antarmuka pengguna yang interaktif dengan standar kinerja yang dioptimalkan. Keunggulan utama dari Next.js terletak pada kemampuannya untuk melakukan *Server-Side Rendering* (SSR) dan *Static Site Generation (SSG)*, yang berfungsi untuk mempercepat waktu muat halaman web secara signifikan. Melalui integrasi yang tepat, aplikasi web yang dibangun dengan *Next.js* dapat dibungkus menggunakan *Capacitor* sehingga bertransformasi menjadi aplikasi mobile yang utuh dan fungsional.

### 2.3.5 Visual Studio Code

Visual Studio Code merupakan *code editor* yang ringan namun fungsional yang dikembangkan oleh Microsoft. Dalam penelitian ini, Visual Studio Code dipilih sebagai lingkungan pengembangan utama untuk merancang dan mengimplementasikan aplikasi deteksi penyakit buah apel. Pemilihan ini didasarkan pada fleksibilitas tinggi serta dukungan ekosistem ekstensi yang luas, yang penting untuk pengembangan aplikasi lintas platform. Visual Studio Code memberikan dukungan penuh terhadap pengembangan aplikasi menggunakan kerangka kerja Next.js dan Capacitor melalui fitur IntelliSense, pelengkapan kode otomatis, dan terminal yang terintegrasi. Dalam konteks integrasi *deep learning*, lingkungan ini memudahkan pengembangan dan menjalankan model ONNX melalui lingkungan JavaScript atau TypeScript.

### 2.3.6 T4-GPU

Dalam pelatihan model *deep learning modern*, *Graphics Processing Unit (*GPU) menjadi komponen vital karena mampu mengeksekusi ribuan operasi matematika secara paralel. Kecepatan ini dibutuhkan saat melatih arsitektur besar berbasis *transfer learning* yang melibatkan jutaan parameter. Salah satu

GPU yang banyak digunakan di lingkungan komputasi awan (seperti Google Colaboratory) adalah NVIDIA Tesla T4, yang dirancang khusus untuk inferensi dan pelatihan berbasis mixed *precision*.

Keunggulan T4 terletak pada *Tensor Cores*, unit khusus yang dioptimalkan untuk operasi perkalian matriks pada presisi campuran (FP16/FP32). Dengan arsitektur ini, model *transfer learning* yang menggunakan *backbone* seperti MobileNetV3 dapat diproses hingga dua kali lebih cepat tanpa kehilangan akurasi, karena *framework* modern menerapkan mekanisme loss scaling otomatis. Kapasitas memori 16 GB juga memastikan dataset citra dengan ukuran 224×224 piksel beserta batch berukuran sedang dapat dimuat secara penuh, mengurangi over*head* transfer data antara CPU dan GPU

### 2.3.7 ONNX (*Open Neural Network Exchange*)

*Open Neural Network Exchange*, atau yang lebih dikenal sebagai ONNX, merupakan sebuah standar terbuka yang dirancang untuk memfasilitasi interoperabilitas antar berbagai kerangka kerja machine learning dan deep learning. Dikembangkan secara kolaboratif oleh perusahaan teknologi besar seperti Microsoft dan Facebook, ONNX bertindak sebagai jembatan yang memungkinkan sebuah model buatan dipindahkan dan dijalankan lintas ekosistem tanpa kendala kompatibilitas. Melalui format representasi yang seragam ini, pelatiha model dapat dilakukan dengan menggunakan satu kerangka kerja tertentu, seperti PyTorch, kemudian mengekspor dan menjalankannya pada kerangka kerja lain, seperti TensorFlow atau Caffe2, tanpa perlu melakukan pelatihan ulang dari awal. Fleksibilitas ini secara efektif mengeliminasi ketergantungan pada satu ekosistem vendor tertentu, sehingga membebaskan peneliti untuk memilih alat terbaik yang paling sesuai untuk setiap tahapan pengembangan sistem.

Selain aspek fleksibilitas, ONNX juga berfokus pada optimalisasi performa inferensi melalui ekosistem pendukungnya yang disebut ONNX *Runtime*. Ketika sebuah model dikonversi ke dalam format ONNX, *runtime* ini akan mengevaluasi graf komputasi model dan menerapkan berbagai teknik optimasi, seperti pemangkasan node yang tidak efisien dan alokasi memori yang lebih dinamis. Hasilnya, model dapat dieksekusi dengan kecepatan tinggi dan efisiensi optimal di atas berbagai arsitektur perangkat keras, mulai dari prosesor konvensional hingga akselerator grafis khusus. Karakteristik inilah yang menjadikan ONNX sebagai solusi dalam menjembatani fase riset yang dinamis dengan fase implementasi produksi yang membutuhkan kestabilan serta performa tinggi.