BAB II
KAJIAN LITERATUR

2.1 Citra

Citra merupakan representasi visual dari suatu objek nyata yang terbentuk melalui interaksi antara cahaya dan objek tersebut. Proses pembentukan citra berlangsung ketika sumber cahaya menerangi suatu objek, lalu objek tersebut memantulkan kembali sebagian berkas cahaya yang diterimanya, dan pantulan cahaya inilah yang kemudian ditangkap oleh alat-alat optik sehingga bayangan objek tersebut terekam menjadi sebuah citra. Ditinjau dari sudut pandang matematis, citra yang dihasilkan dapat dimodelkan sebagai fungsi menerus dari intensitas cahaya pada bidang dua dimensi. Karena memuat informasi visual yang jauh lebih kaya dibandingkan data dalam bentuk teks, citra menjadi bentuk representasi yang memungkinkan informasi dari dunia nyata direkam dan dimanfaatkan sebagai bahan bagi berbagai keperluan analisis [16].

Citra dapat dibedakan menjadi dua jenis, yaitu citra kontinu dan citra diskrit. Citra kontinu dihasilkan oleh sistem optik yang menerima sinyal analog, seperti mata manusia dan kamera analog, sedangkan citra diskrit yang juga dikenal sebagai citra digital diperoleh melalui proses digitalisasi terhadap citra kontinu. Sebagian sistem optik telah dilengkapi fungsi digitalisasi sehingga mampu langsung menghasilkan citra digital, seperti kamera digital dan pemindai (*scanner*). Karena komputer digital yang umum digunakan saat ini hanya mampu mengolah citra digital, bentuk citra digital inilah yang menjadi format lazim digunakan dalam berbagai penelitian berbasis komputasi [17].

Sebuah citra membawa informasi visual yang dapat diuraikan menjadi sejumlah fitur penyusunnya. Fitur-fitur tersebut antara lain berupa tepi, bentuk, tekstur, warna, dan pola, yang masing-masing dapat dikenali maupun diekstraksi sebagai dasar bagi proses analisis citra. Selain kandungan fiturnya, kualitas sebuah citra turut ditentukan oleh resolusinya, yaitu jumlah piksel yang menyusun citra tersebut. Citra dengan resolusi yang lebih tinggi memuat lebih banyak detail karena tersusun atas jumlah piksel yang lebih banyak, sedangkan citra dengan resolusi rendah cenderung lebih kasar sehingga lebih sulit diinterpretasi [18].

2.1.1 Jenis-jenis Citra

Citra digital dapat diklasifikasikan berdasarkan nilai yang direpresentasikan pada setiap piksel penyusunnya. Perbedaan nilai tersebut menentukan jenis informasi visual yang

dapat disajikan, sehingga setiap jenis citra memiliki karakteristik dan kegunaan yang berbeda. Berdasarkan ketentuan tersebut, citra digital secara umum terbagi menjadi tiga jenis, yaitu citra berwarna, citra berskala keabuan (*grayscale*), dan citra biner [19]. Ketiga jenis citra tersebut diuraikan pada sebagai berikut.

1. Citra Berwarna

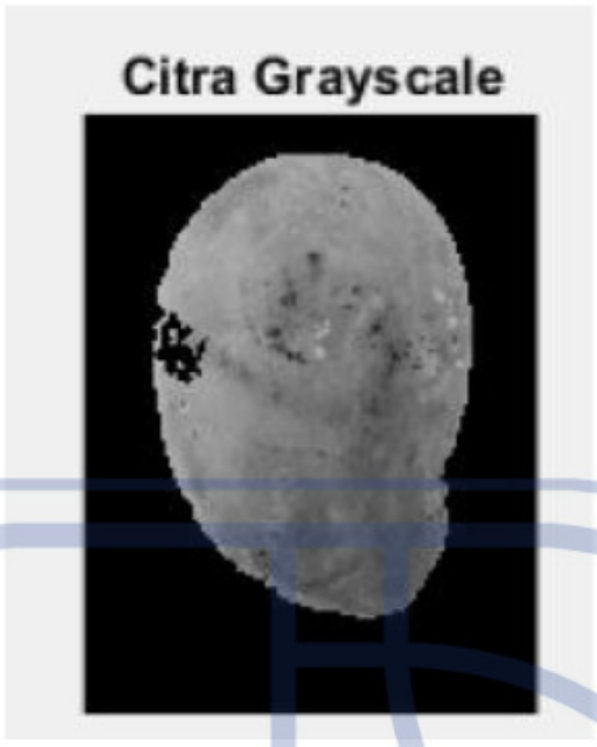
   Citra berwarna merupakan jenis citra yang menyajikan informasi warna melalui kombinasi beberapa komponen warna pada setiap pikselnya. Setiap piksel pada citra berwarna tidak hanya menyimpan satu nilai intensitas, melainkan beberapa nilai yang secara bersama-sama membentuk warna tertentu. Jenis citra ini umum dikenal dengan istilah *truecolor* atau citra RGB, yang tersusun atas tiga komponen warna utama, yaitu merah (*red*), hijau (*green*), dan biru (*blue)*. Kombinasi dari ketiga komponen tersebut menghasilkan variasi warna yang luas sehingga dapat merepresentasikan warna objek mendekati kondisi aslinya. Citra berwarna digunakan ketika informasi warna menjadi atribut penting dalam proses pengamatan dan analisis objek [19].

Gambar 2.1 Citra RGB [20]

2. Citra Berskala Keabuan *(Grayscale)*

   Citra berskala keabuan merupakan jenis citra yang hanya merepresentasikan tingkat intensitas cahaya tanpa melibatkan informasi warna. Setiap piksel pada citra ini menyimpan satu nilai numerik yang menggambarkan gradasi dari gelap hingga terang. Nilai terendah menyatakan warna hitam, sedangkan nilai tertinggi menyatakan warna putih, dengan nilai di antaranya membentuk berbagai tingkat keabuan. Variasi intensitas tersebut menyusun struktur visual citra dalam bentuk gradasi keabuan yang berkesinambungan. Citra berskala keabuan digunakan ketika analisis difokuskan pada

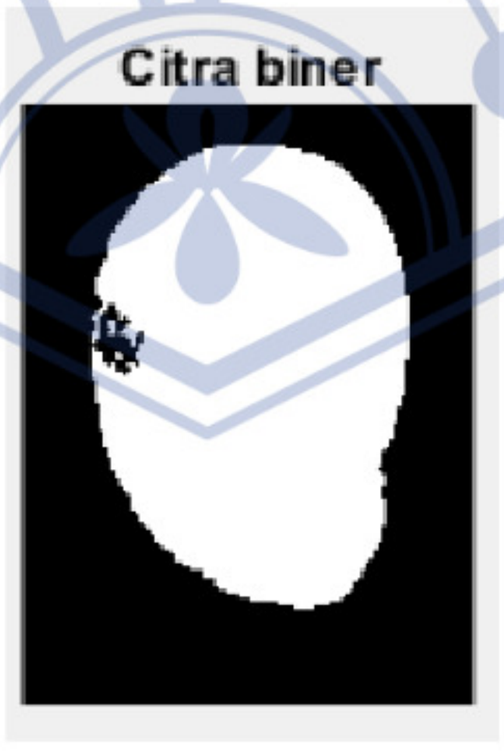
distribusi intensitas, sehingga kebutuhan komputasi dapat ditekan dibanding pemrosesan citra berwarna [19].



Gambar 2.2 Citra Berskala Keabuan *(Grayscale)* [20]

3. Citra Biner

Citra biner merupakan jenis citra yang hanya memiliki dua kemungkinan nilai pada setiap pikselnya. Kedua nilai tersebut merepresentasikan dua kondisi visual yang kontras, yaitu objek dan latar belakang, yang umumnya dinyatakan sebagai hitam dan putih. Struktur yang sederhana menjadikan citra biner sebagai bentuk citra yang paling efisien dari sisi penyimpanan maupun pemrosesan. Penggunaannya umum ditemukan pada situasi yang memerlukan pemisahan objek secara jelas dari latar belakangnya, seperti pengenalan teks pada halaman dan deteksi siluet objek [19].



Gambar 2.3 Citra Biner [20]

2.1.2 Representasi Citra

Secara matematis, citra digital direpresentasikan sebagai sebuah fungsi dua dimensi $f(x, y)$, dengan x dan y menyatakan koordinat spasial dan nilai fungsi pada setiap titik menyatakan intensitas citra pada posisi tersebut. Representasi yang semula kontinu tersebut diubah menjadi representasi diskrit melalui proses digitalisasi, yaitu pencuplikan (*sampling*) terhadap koordinat spasial citra sehingga citra terbagi menjadi sejumlah baris dan kolom [19]. Hasil dari proses ini menyimpan informasi citra dalam bentuk nilai-nilai numerik pada setiap titiknya sehingga citra dapat dipahami dan diproses oleh computer [18].

Dalam bentuk diskritnya, citra digital tersusun atas elemen-elemen terkecil yang disebut piksel dan secara keseluruhan membentuk sebuah matriks dua dimensi [18]. Setiap piksel menempati posisi tertentu yang dinyatakan dalam koordinat baris (m) dan kolom (n), dengan perpotongan antara baris dan kolom tersebut merepresentasikan satu piksel [19]. Representasi dalam bentuk matriks ini memungkinkan citra dimanipulasi melalui operasi-operasi matematis (operasi matriks), sementara dimensi atau ukuran citra itu sendiri dinyatakan dalam jumlah baris dan kolom pikselnya [18], [19]. Nilai yang terkandung pada setiap piksel inilah yang pada akhirnya menentukan informasi visual yang direpresentasikan oleh citra.

Nilai yang tersimpan pada setiap piksel merepresentasikan tingkat intensitas kecerahan ataupun informasi warna pada titik tersebut. Pada citra berkanal tunggal atau citra skala keabuan (*grayscale*), setiap piksel cukup disimpan dalam satu nilai numerik, sedangkan pada citra multikanal atau citra berwarna setiap piksel disimpan dalam beberapa nilai sekaligus, misalnya tiga nilai untuk model warna RGB (*Red, Green, dan Blue*). Nilai-nilai tersebut umumnya berada pada rentang 0 hingga 255 untuk representasi 8-bit, dengan nilai 0 menyatakan intensitas paling rendah (hitam) dan nilai 255 menyatakan intensitas paling tinggi (putih) pada citra skala keabuan, atau menyatakan besarnya komponen warna pada citra berwarna. Variasi nilai antarpiksel inilah yang membentuk tampilan visual citra sekaligus menentukan kebutuhan penyimpanannya, sehingga ketepatan dalam merepresentasikan informasi warna menjadi penting bagi penelitian yang mengandalkan perubahan warna sebagai penanda kondisi objek [18].

2.2 Klasifikasi Citra

Klasifikasi citra merupakan proses pemberian label atau kategori tertentu pada suatu citra digital berdasarkan informasi visual yang terkandung di dalamnya. Dalam bidang *computer vision*, klasifikasi citra menjadi salah satu permasalahan utama yang digunakan

untuk mengenali dan membedakan objek atau kategori tertentu dari gambar digital [21]. Proses klasifikasi citra umumnya dilakukan dengan mengekstraksi fitur dari citra, kemudian menggunakan model pembelajaran mesin untuk mempelajari pola dari data citra tersebut sehingga citra dapat dikelompokkan ke dalam kelas yang sesuai [22]. Dengan kemampuan tersebut, klasifikasi citra banyak digunakan dalam berbagai aplikasi pengolahan citra dan sistem pengenalan visual untuk membantu komputer memahami dan menginterpretasikan informasi yang terdapat pada gambar.

Pada awal perkembangannya, metode klasifikasi citra banyak menggunakan pendekatan tradisional yang bergantung pada proses ekstraksi fitur secara manual (*handcrafted features*). Dalam pendekatan ini, fitur citra dirancang secara khusus oleh peneliti atau pakar berdasarkan karakteristik tertentu dari data yang dianalisis, sehingga proses klasifikasi sangat bergantung pada pengetahuan sebelumnya mengenai objek yang akan dikenali. Pendekatan tersebut memiliki keterbatasan dalam menangani data citra berukuran besar karena proses perancangan fitur secara manual cenderung kurang efisien dan sulit mengakomodasi kompleksitas data visual yang semakin meningkat [23]. Seiring berkembangnya teknologi *Deep learning*, pendekatan klasifikasi citra mengalami kemajuan signifikan melalui penggunaan *Convolutional Neural Network* (CNN), yang saat ini menjadi salah satu metode utama dalam penelitian klasifikasi citra. CNN mampu mempelajari representasi fitur secara otomatis dari data citra melalui jaringan saraf berlapis, sehingga proses ekstraksi fitur tidak lagi dilakukan secara manual. Melalui struktur jaringan yang terdiri dari beberapa lapisan seperti lapisan konvolusi dan *pooling*, CNN dapat mengekstraksi fitur visual secara hierarkis mulai dari pola sederhana hingga pola yang lebih kompleks, sehingga mampu meningkatkan performa sistem klasifikasi citra pada berbagai aplikasi pengolahan citra dan pengenalan objek [24].

Kinerja model dalam klasifikasi citra umumnya dievaluasi menggunakan beberapa metrik evaluasi yang dihitung melalui analisis confusion matrix. Matriks ini terdiri dari empat komponen utama yaitu *True Positive* (TP), *False Positive* (FP), *True Negative* (TN), dan *False Negative* (FN) yang digunakan untuk mengukur seberapa baik model dalam mengklasifikasikan data ke dalam kelas yang benar. Berdasarkan komponen tersebut, berbagai metrik evaluasi seperti *Accuracy*, *Precision*, *Recall*, dan *F1-Score* dapat dihitung untuk menilai performa model secara lebih komprehensif [25]. Dalam praktiknya, proses klasifikasi citra juga dapat dipengaruhi oleh berbagai faktor seperti variasi pencahayaan, perbedaan sudut pengambilan gambar, serta perubahan warna atau tekstur objek yang dapat memengaruhi hasil klasifikasi. Pada penelitian ini, klasifikasi citra digunakan untuk

mengidentifikasi tingkat kematangan buah mangga berdasarkan karakteristik visual yang terdapat pada citra buah, khususnya informasi warna dan tekstur yang menjadi indikator penting dalam membedakan tingkat kematangan buah tersebut.

2.3 Pengolahan Citra Digital

Pengolahan citra digital mencakup penerapan berbagai operasi terhadap citra dalam bentuk digital untuk menghasilkan keluaran tertentu. Dalam proses ini, citra berfungsi sebagai data masukan yang diproses secara komputasional, dengan keluaran berupa citra baru atau informasi yang merepresentasikan karakteristik citra tersebut. Secara konseptual, pengolahan citra merupakan bagian dari pemrosesan sinyal dua dimensi, di mana setiap nilai piksel diperlakukan sebagai komponen sinyal yang dapat dianalisis dan dimodifikasi melalui pendekatan matematis [18], [19].

Tujuan pengolahan citra digital meliputi peningkatan kualitas visual serta perolehan informasi dari citra. Peningkatan kualitas dilakukan melalui penyesuaian terhadap aspek kecerahan, kontras, dan pengurangan derau (*noise*) untuk memperjelas informasi yang terkandung. Selain itu, pengolahan citra digunakan untuk mengekstraksi karakteristik penting seperti pola, bentuk, atau fitur objek, sehingga citra dapat dimanfaatkan sebagai sumber data dalam proses analisis [18].

Pengolahan citra melibatkan berbagai operasi yang bekerja pada nilai piksel untuk menghasilkan perubahan tertentu pada citra tanpa mengubah representasi dasarnya. Penerapannya dapat ditemukan pada berbagai bidang, seperti analisis citra medis dan sistem pengawasan berbasis visual yang memanfaatkan informasi citra untuk tujuan tertentu. Dalam implementasinya, pengolahan citra mencakup sejumlah tahapan dasar sebelum analisis lanjutan, seperti penyesuaian ukuran citra, normalisasi nilai, dan pengayaan data, yang akan dibahas lebih lanjut pada subbab berikutnya [18], [19].

2.3.1 *Resize*

Resize merupakan prosedur transformasi geometris fundamental dalam pra-pemrosesan citra digital yang memanipulasi resolusi spasial untuk menghasilkan representasi visual baru dengan dimensi yang diperbesar (*upscaling*) atau diperkecil (*downscaling*) sesuai dengan kebutuhan sistem analisis [26]. Dalam kerangka kerja Deep Learning seperti arsitektur EfficientNet-B0, standardisasi dimensi melalui interpolasi citra sangat krusial karena algoritma ini menyerap ke dalam berbagai aplikasi pemrosesan untuk menghasilkan estimasi nilai piksel yang realistis pada grid matriks input yang seragam [27].

Ditinjau dari aspek operasional, penerapan teknik *downscaling* mampu mereduksi beban komputasi secara signifikan, meskipun kompresi dimensi yang terlalu ekstrem berisiko mendegradasi ketajaman informasi visual yang secara kuantitatif direpresentasikan oleh penurunan nilai Peak Signal-to-Noise Ratio (PSNR) [26]. Guna mengompensasi potensi distorsi pada fitur visual utama seperti bentuk dan gradasi warna kematangan buah mangga, teknik interpolasi konvensional seperti Bicubic diimplementasikan untuk menyusun ulang piksel target berdasarkan perhitungan skema pembobotan area (*weighting schemes*) dari piksel-piksel tetangga [27]. Selanjutnya terdapat rumus perhitungan faktor skala dalam proses *resize*:

$$(\text{Scale}_x, \text{Scale}_y) = \left(\frac{\text{Wo}}{\text{Wt}}, \frac{\text{Ho}}{\text{Ht}}\right) \quad (2.1)$$

Dimana:

$Scale_x$ = faktor skala untuk lebar citra.

$Scale_y$ = faktor skala untuk tinggi citra.

W_t = lebar citra target (*target width*).

H_t = tinggi citra target (*target height*).

W_o = lebar asli citra sebelum *resize* (*original width*).

H_o = tinggi asli citra sebelum *resize* (*original height*).

2.3.2 Normalisasi

Normalisasi citra merupakan proses penyesuaian nilai intensitas piksel ke dalam rentang tertentu untuk menyelaraskan skala data pada citra digital. Proses ini memetakan nilai piksel ke skala yang seragam sehingga perbedaan intensitas akibat kondisi akuisisi atau karakteristik citra dapat diminimalkan. Dengan demikian, citra yang memiliki variasi rentang nilai dapat diselaraskan sehingga mendukung konsistensi dalam tahap pengolahan selanjutnya [19], [28].

Normalisasi citra dilakukan melalui transformasi matematis terhadap nilai piksel menggunakan pendekatan tertentu. Salah satu metode yang umum digunakan adalah *Scaling normalization*, yaitu memetakan nilai intensitas dari rentang awal, seperti 0–255, ke rentang baru seperti 0–1. Selain itu, *Z-Score normalization* menstandarkan nilai piksel berdasarkan rata-rata dan simpangan baku sehingga menghasilkan distribusi dengan rata-rata nol dan variansi satu. Perbedaan kedua metode tersebut terletak pada prinsip transformasinya, di mana penskalaan bersifat linier, sedangkan *Z-Score* mempertimbangkan karakteristik statistik data [28].

Dalam rangkaian pengolahan citra digital, normalisasi citra berperan sebagai bagian dari tahap prapemrosesan yang menyiapkan data sebelum analisis lanjutan. Penyesuaian nilai intensitas melalui normalisasi mendukung konsistensi representasi data masukan serta mempermudah penerapan metode yang memerlukan rentang nilai tertentu. Dengan demikian, normalisasi citra menjadi tahap penting dalam memastikan kesiapan data sebelum diterapkan teknik pengolahan lanjutan lainnya [18], [28].

Penjelasan dari teori diatas menyatakan bahwa terdapat rumus normalisasi untuk menghitung rentang 0 hingga 1:

$$X = \frac{x}{255} \quad \text{..... (2.2)}$$

Dimana:

X = nilai piksel citra setelah proses normalisasi.

255 = nilai intensitas piksel maksimum pada citra 8-bit.

Selanjutnya terdapat rumus Normalisasi *Z-Score* dalam perhitungan normalisasi penjelasan diatas:

Rumus Normalisasi *Z-Score*:

$$X_{norm} = \frac{X-\mu}{\sigma} \quad \text{..... (2.3)}$$

Dimana:

X_{norm} = nilai piksel hasil standardisasi (normalisasi *z-score*).

X = nilai piksel citra setelah proses normalisasi.

μ = nilai rata-rata (mean) dari data.

Σ = simpangan baku (*standard deviation*) dari data.

2.3.3 Augmentasi Data

Augmentasi data dilakukan melalui modifikasi citra untuk menghasilkan variasi dari data yang telah tersedia dengan tetap mempertahankan karakteristik utama objek. Proses ini melibatkan penerapan transformasi tertentu sehingga diperoleh representasi data yang lebih beragam tanpa memerlukan akuisisi citra tambahan. Dalam pengolahan citra digital, augmentasi diterapkan langsung pada data visual, sehingga informasi yang terkandung tetap dapat digunakan dalam proses analisis. Pendekatan ini digunakan untuk memperluas representasi data melalui variasi yang dihasilkan secara artifisial [6], [13].

Pada tugas computer vision untuk pengenalan objek alami, seperti klasifikasi tingkat kematangan buah mangga, citra yang ditangkap sering kali memiliki variasi orientasi, posisi,

skala, dan kondisi pencahayaan yang tidak menentu. Oleh karena itu, memodifikasi citra melalui metode augmentasi menjadi tahapan yang sangat penting untuk menyimulasikan berbagai kondisi pengambilan gambar di dunia nyata. Melalui paparan variasi data ini, model dipaksa untuk benar-benar memahami karakteristik visual intrinsik dari objek, seperti tekstur permukaan dan perubahan distribusi warna pada kulit buah yang menandakan tingkat kematangannya alih-alih sekadar menghafal posisi piksel yang statis pada gambar pelatihan [6], [13].

Dalam rangkaian pengolahan citra digital, augmentasi data berfungsi sebagai bagian dari tahap prapemrosesan yang menyiapkan data sebelum analisis dilakukan. Penambahan variasi citra membantu menjaga konsistensi distribusi data masukan serta mendukung kestabilan proses analisis terhadap variasi kondisi citra. Dengan tersedianya data yang lebih beragam, sistem memiliki representasi yang lebih memadai terhadap objek yang diamati. Oleh karena itu, augmentasi data menjadi salah satu tahap penting dalam mempersiapkan data visual untuk proses pengolahan selanjutnya [13]. Berdasarkan kebutuhan untuk menyimulasikan variasi kondisi citra secara optimal, metode augmentasi data yang diterapkan adalah sebagai berikut:

1. *Rotation*: Teknik ini memutar gambar secara acak dengan rentang sudut tertentu, baik searah maupun berlawanan arah jarum jam. Transformasi ini melatih model agar kebal (*invariant*) terhadap perubahan orientasi objek, sehingga posisi buah yang miring pada gambar tidak memengaruhi hasil klasifikasi [6].
2. *Shift*: Memindahkan posisi objek di dalam citra secara horizontal atau vertikal dalam persentase tertentu. Teknik ini bertujuan mencegah model menjadi bias terhadap objek yang selalu berada di tengah bingkai, merepresentasikan variasi letak objek pada kondisi penangkapan gambar yang sebenarnya [6].
3. *Zoom*: Mengubah ukuran gambar dengan memperbesar atau memperkecil tampilannya secara acak. Manipulasi ini membantu model melakukan generalisasi dengan lebih baik dengan menyimulasikan variasi jarak pengambilan citra antara lensa kamera dan objek buah [13].
4. *Flipping*: Teknik augmentasi yang melakukan pembalikan struktur citra secara simetris, baik secara horizontal maupun vertical. Teknik ini secara efektif menggandakan variasi sudut pandang objek secara simetris, memperkuat pengenalan model terhadap pola bentuk objek tanpa mengubah representasi kelas aslinya [13].
5. *Brightness Adjustment*: Teknik yang secara dinamis mengubah tingkat intensitas cahaya pada citra. Penyesuaian ini memungkinkan model untuk beradaptasi dan lebih tahan

(*robust*) terhadap variasi kondisi pencahayaan lingkungan, baik saat kondisi terlalu terang (*overexposed*) maupun redup [13].

6. *Channel Shifting*: Menambahkan nilai acak secara independen pada saluran warna RGB citra. Manipulasi ini bertujuan menyimulasikan perbedaan kalibrasi sensor kamera atau variasi saturasi warna alami yang terjadi pada kulit buah [29].

Secara matematis, penentuan jumlah gambar baru yang didapatkan untuk setiap metode augmentasi dirumuskan sebagai berikut:

$$\text{Jumlah augmentasi per teknik} = \frac{\text{Total gambar tambahan}}{\text{Jumlah teknik augmentasi}} \quad \text{..... (2.4)}$$

2.3.4 Ekstraksi Fitur

Ekstraksi fitur merupakan proses pengambilan informasi dari citra untuk menghasilkan representasi yang lebih ringkas dengan tetap mempertahankan karakteristik utama objek. Dalam proses ini, data citra yang tersusun atas nilai piksel diubah menjadi sekumpulan fitur dalam bentuk numerik yang mencerminkan karakteristik visual tertentu. Transformasi tersebut memungkinkan penyederhanaan struktur data citra sehingga lebih efisien untuk dianalisis tanpa menghilangkan informasi yang relevan [19].

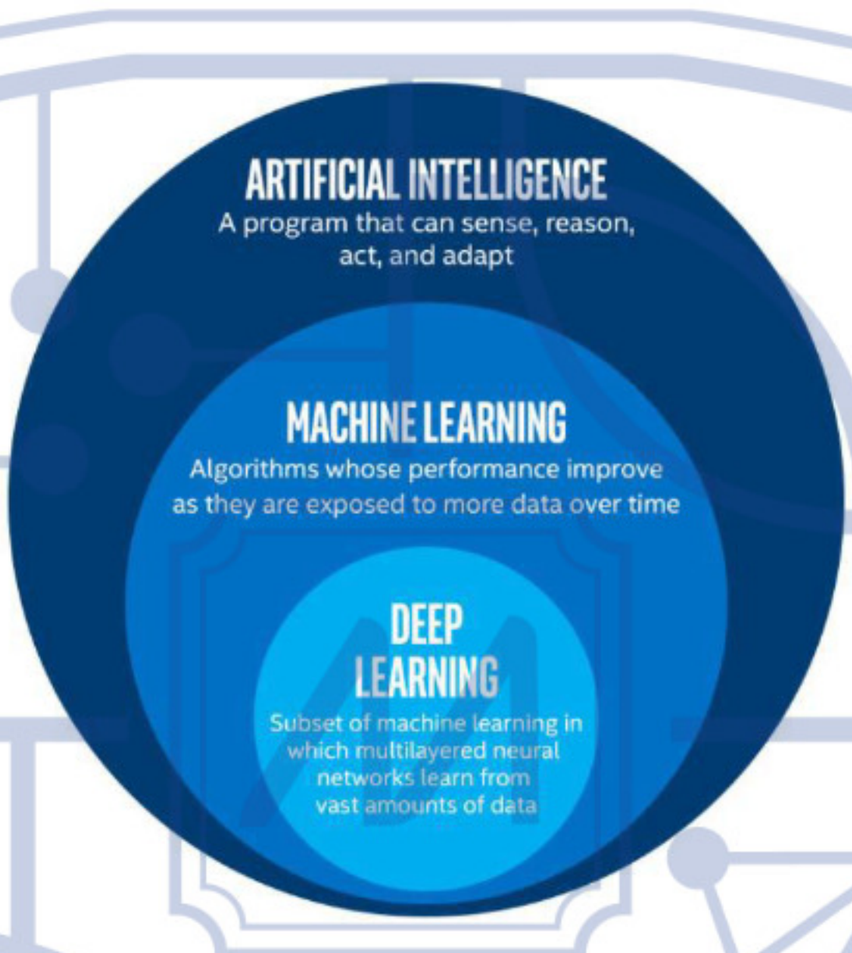
Karakteristik yang diperoleh dari citra umumnya meliputi warna, tekstur, dan bentuk objek. Fitur warna menggambarkan distribusi intensitas warna, sedangkan fitur tekstur merepresentasikan pola atau susunan intensitas yang berulang pada suatu permukaan. Adapun fitur bentuk berkaitan dengan karakteristik geometris objek, seperti kontur dan area. Ketiga jenis fitur tersebut memberikan representasi yang saling melengkapi dalam mendeskripsikan objek yang terdapat dalam citra [18], [19].

Dalam alur pengolahan citra digital, ekstraksi fitur dilakukan setelah tahap prapemrosesan untuk menghasilkan data yang lebih terstruktur dalam bentuk fitur. Representasi ini memungkinkan citra diolah sebagai sekumpulan karakteristik yang lebih sederhana dibandingkan dengan data piksel mentah. Dengan demikian, ekstraksi fitur mendukung proses analisis terhadap pola dan karakteristik objek, serta berperan sebagai penghubung antara data citra dan tahap analisis selanjutnya [18], [19].

2.4 *Deep Learning*

Deep Learning merupakan cabang dari pembelajaran mesin (*Machine Learning*) yang memanfaatkan jaringan saraf tiruan berlapis (*deep neural network*) untuk mempelajari representasi data secara otomatis. Pendekatan ini berbeda dari metode tradisional yang

mengandalkan rekayasa fitur secara manual, karena model mampu mengekstraksi pola langsung dari data mentah. Kemampuan tersebut memungkinkan identifikasi struktur kompleks pada berbagai jenis data, termasuk citra, suara, dan teks. Proses pembelajaran dalam *Deep Learning* bersifat hierarkis, di mana setiap lapisan membentuk representasi dengan tingkat abstraksi yang meningkat secara bertahap [30]. Hubungan hierarkis antara *Artificial Intelligence, Machine Learning,* dan *Deep Learning* ditunjukkan secara visual pada Gambar 2.4



Gambar 2.4 *Deep Learning* [31]

Representasi tersebut diperoleh melalui pemrosesan data secara berlapis dalam jaringan saraf tiruan yang terdiri dari lapisan masukan (*input Layer*), lapisan tersembunyi (*hidden Layer*), dan lapisan keluaran (*output Layer*). Setiap lapisan tersusun atas *neuron* yang saling terhubung melalui parameter berupa bobot (*weight*) dan bias. Selama proses komputasi, data mengalami transformasi bertahap dari bentuk awal menuju representasi yang lebih abstrak melalui kombinasi operasi linear dan fungsi aktivasi *non-linear*. Fungsi aktivasi berperan dalam memungkinkan model menangkap hubungan *non-linear* dalam data. Proses pelatihan dilakukan dengan menyesuaikan parameter berdasarkan data pelatihan sehingga kesalahan prediksi dapat diminimalkan [30].

Kemampuan dalam membentuk representasi hierarkis tersebut menjadikan *Deep Learning* relevan untuk analisis data visual. Dalam pengolahan citra, pendekatan ini digunakan untuk mengekstraksi pola visual secara langsung tanpa tahapan rekayasa fitur secara manual. Penerapannya mencakup berbagai tugas, seperti klasifikasi citra, deteksi objek, dan pengenalan pola. Karakteristik ini mendasari penggunaan *Deep Learning* sebagai

pendekatan utama dalam pemrosesan citra, sekaligus menjadi landasan bagi pengembangan metode yang lebih spesifik [30].

2.5 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) merupakan salah satu model jaringan saraf tiruan yang dirancang untuk memproses data berbentuk matriks, seperti citra. Berbeda dengan pendekatan konvensional yang memerlukan proses ekstraksi fitur secara manual, CNN memiliki kemampuan untuk melakukan pembelajaran *end-to-end*. Dengan pendekatan ini, model dapat secara otomatis mempelajari pola dari data mentah berupa piksel gambar dan mengonversinya menjadi representasi fitur tingkat tinggi yang digunakan dalam proses klasifikasi objek [32]. Secara umum, arsitektur CNN terdiri atas beberapa komponen utama yang masing-masing memiliki fungsi tertentu, yaitu sebagai berikut:

1. *Convolution Layer*

   Lapisan konvolusi merupakan komponen utama yang berfungsi untuk mengekstraksi fitur atau pola dari area lokal pada citra masukan. Proses ini dilakukan dengan menggeser filter atau kernel pada seluruh bagian gambar. Pada setiap pergeseran, dilakukan operasi perkalian antara nilai filter dan nilai piksel yang bersesuaian, kemudian hasilnya dijumlahkan untuk membentuk *feature map* [32]. Mekanisme ini memungkinkan penggunaan parameter yang lebih efisien, karena satu filter dapat digunakan untuk mendeteksi pola pada seluruh area citra.

2. *Activation Function*

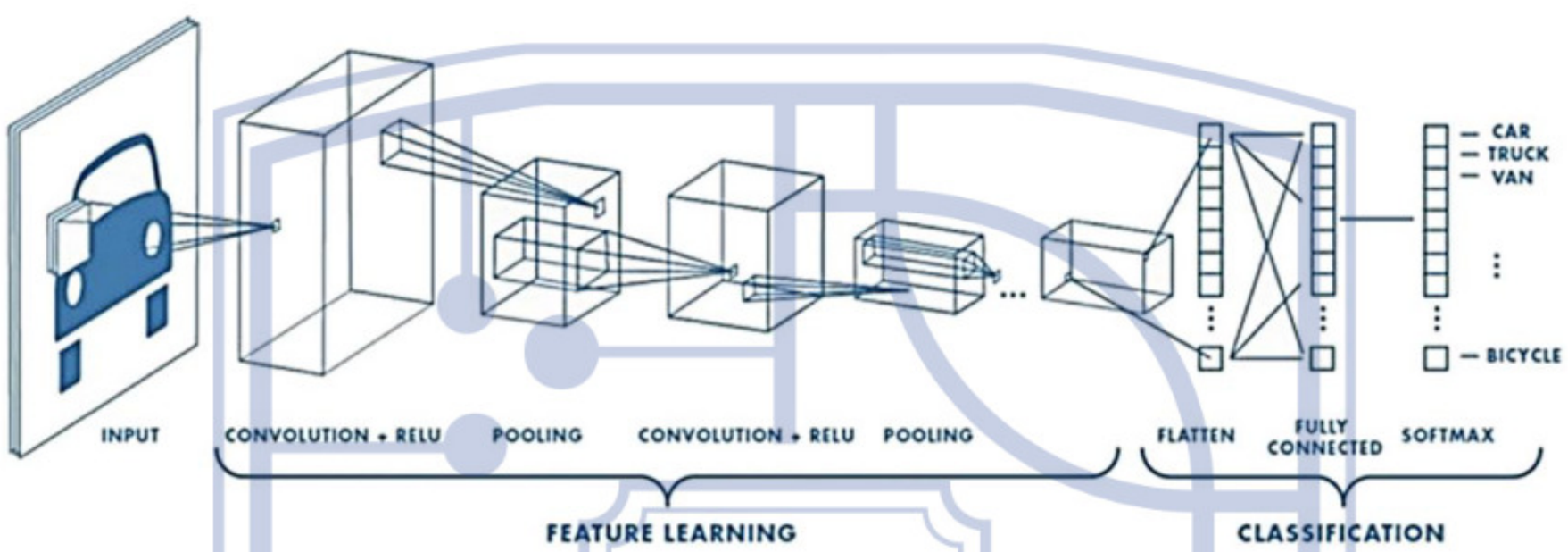
   Fungsi aktivasi berperan dalam memberikan sifat *non-linear* pada jaringan saraf, sehingga model mampu mempelajari pola yang kompleks dalam data. Dengan adanya fungsi ini, jaringan tidak hanya melakukan transformasi linier, tetapi juga dapat menangkap hubungan yang lebih rumit, sehingga lebih efektif dalam mengenali objek pada data dunia nyata [32].

3. *Pooling Layer*

   Lapisan *Pooling* digunakan setelah proses konvolusi untuk mereduksi dimensi *feature map*. Proses ini dilakukan dengan membagi *feature map* ke dalam beberapa region kecil dan mengambil nilai representatif, seperti nilai maksimum pada metode *max pooling*. Selain mengurangi ukuran data dan beban komputasi, *Pooling* juga membantu meningkatkan ketahanan model terhadap variasi kecil pada citra serta mengurangi risiko *overfitting* [32].

4. *Fully connected Layer*

Lapisan terhubung penuh umumnya terletak pada bagian akhir arsitektur CNN dan berfungsi untuk menghasilkan keputusan akhir atau klasifikasi. Berbeda dengan lapisan sebelumnya yang beroperasi secara lokal, pada lapisan ini setiap *neuron* terhubung dengan seluruh *neuron* pada lapisan sebelumnya. Lapisan ini menggabungkan seluruh fitur yang telah diekstraksi menjadi satu representasi utuh untuk menentukan kategori akhir dari citra yang diproses [32].



Gambar 2.5 *Convolutional Neural Network Architecture* [31]

2.5.1 *Forward pass*

Forward pass merupakan tahap komputasi pada *Convolutional Neural Network* (CNN) di mana data diproses secara berurutan dari lapisan awal hingga lapisan akhir. CNN sendiri merupakan jaringan saraf berlapis (*feed-forward neural network*) yang bekerja dengan cara mengalirkan informasi satu arah. Pada proses ini, data berupa citra secara bertahap ditransformasikan dari representasi piksel mentah menjadi representasi fitur tingkat tinggi melalui serangkaian lapisan pemrosesan. Hasil dari proses tersebut kemudian digunakan untuk menghasilkan prediksi akhir pada tahap klasifikasi [32]. Dalam implementasi arsitektur yang digunakan, proses *forward pass* melibatkan beberapa tahapan komputasi utama, mulai dari reduksi dimensi hingga pembentukan *output* prediksi, yang dapat dijelaskan melalui persamaan berikut.

1. *Global Average Pooling (*GAP*)*

   Proses peringkasan fitur spasial dilakukan menggunakan metode *Global Average Pooling (*GAP*)*, yaitu teknik yang merangkum setiap kanal fitur menjadi satu nilai representatif. Metode ini berfungsi untuk mengurangi dimensi data sekaligus mempertahankan informasi penting dari *feature map* [32]. Secara matematis, GAP dapat dituliskan sebagai berikut:

$$Y_c = \frac{1}{H \times W} \sum X_{i,j} \quad \text{..... (2.5)}$$

Dimana:

Y_c = nilai rata-rata dari kanal ke-c

$H \times W$ = ukuran dimensi tinggi dan lebar *feature map*

$X_{i,j}$ = nilai elemen fitur pada posisi spasial (i, j)

2. Perhitungan *Logits* pada *Fully Connected Layer*

Setelah proses peringkasan fitur, data diteruskan ke *fully connected layer* yang bertugas menghasilkan skor klasifikasi. Pada tahap ini, setiap keluaran merupakan hasil kombinasi linear antara *input*, bobot, dan bias [32]. Secara matematis, perhitungan nilai *logits* dapat dirumuskan sebagai berikut:

$$z_k = \sum(x_i \cdot w_{k,i}) + b_k \quad \text{..... (2.6)}$$

Dimana:

z_k = nilai *logits* untuk kelas ke-k

x_i = elemen vektor fitur *input*

$w_{k,i}$ = bobot yang menghubungkan *input* ke *neuron* kelas ke-k

b_k = bias pada kelas ke-k

3. Fungsi Aktivasi *Softmax*

Pada tugas klasifikasi multi-kelas, fungsi aktivasi *Softmax* digunakan untuk mengubah nilai *logits* menjadi distribusi probabilitas. Fungsi ini menghasilkan nilai antara 0 hingga 1, dengan total seluruh probabilitas sama dengan 1, sehingga dapat diinterpretasikan sebagai peluang setiap kelas [32]. Secara matematis, *Softmax* dirumuskan sebagai:

$$y_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{..... (2.7)}$$

Dimana:

y_i = probabilitas prediksi untuk kelas ke-i

z_i = nilai *logits* untuk kelas ke-i

K = jumlah total kelas

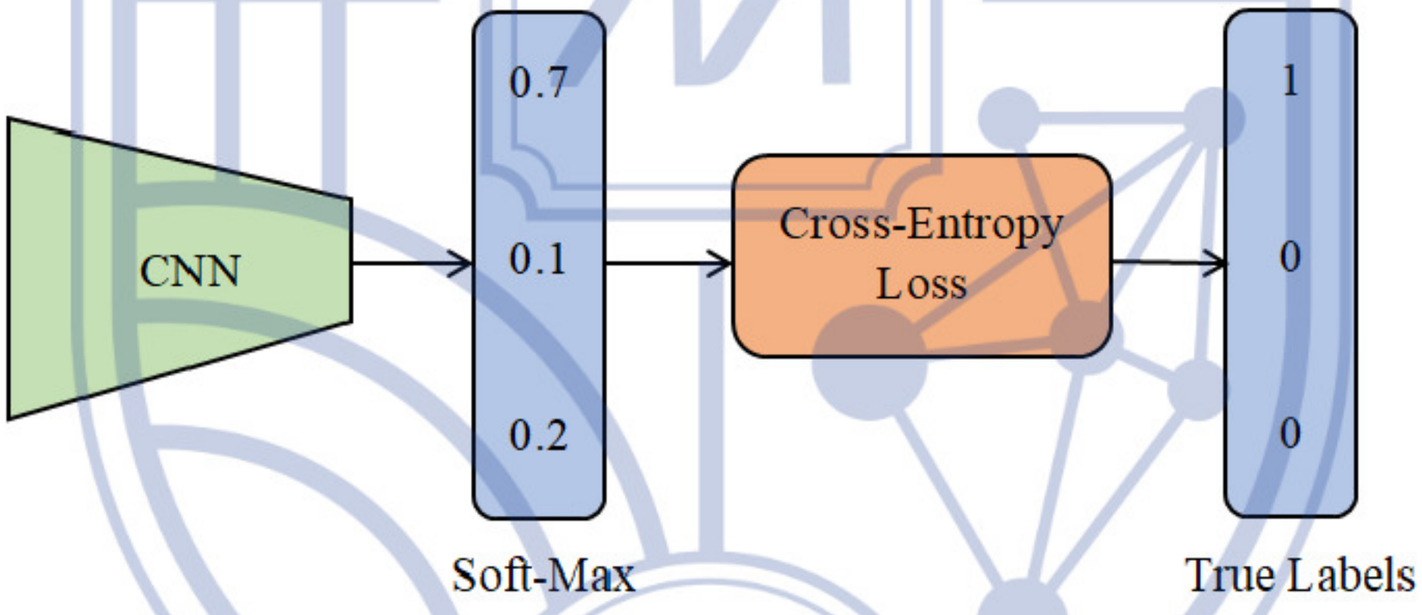
2.5.2 Fungsi *Loss*

Dalam arsitektur *Deep Learning* seperti *Convolutional Neural Network* (CNN), fungsi *loss* merupakan komponen penting yang digunakan pada tahap pelatihan (*training*). Fungsi ini berperan untuk mengukur perbedaan antara hasil prediksi yang dihasilkan oleh model dengan label target yang sebenarnya (*ground truth*) [33]. Secara umum, nilai *loss* digunakan

sebagai indikator kinerja model, di mana nilai yang lebih kecil menunjukkan bahwa prediksi model semakin mendekati nilai sebenarnya. Selain itu, nilai *loss* ini menjadi dasar dalam proses *backpropagation* untuk memperbarui parameter jaringan, sehingga kesalahan dapat diminimalkan dan performa model dapat ditingkatkan.

Pada permasalahan klasifikasi citra dengan banyak kelas (*multi-class classificatio*n), fungsi *loss* yang umum digunakan adalah *Categorical Cross Entropy* (CCE). Fungsi ini dirancang khusus untuk menangani data kategorikal dan mengukur perbedaan antara distribusi probabilitas hasil prediksi model dengan distribusi label sebenarnya. Oleh karena itu, CCE sangat sesuai digunakan pada model yang menghasilkan *output* probabilitas, seperti yang dihasilkan oleh fungsi aktivasi *Softmax* [33].

Secara prinsip, CCE bekerja dengan membandingkan probabilitas prediksi terhadap label target, kemudian memberikan penalti yang lebih besar pada prediksi yang menyimpang dari nilai sebenarnya. Dengan demikian, model didorong untuk menghasilkan probabilitas yang tinggi pada kelas yang benar dan rendah pada kelas lainnya, sehingga kesalahan klasifikasi dapat diminimalkan [33].



Gambar 2.6 *Cross Entropy in* CNN [34]

Secara matematis, fungsi *Categorical Cross Entropy* dapat dirumuskan sebagai berikut:

Mencari Volume (V) Bola:

$$L_{CE} = -\sum_{i=1}^{K} y_i \log(p_i) \quad \text{..... (2.8)}$$

Dimana:

L_{CE} = nilai *Loss* Categorical Cross Entropy

K = jumlah kelas (pada penelitian ini sebanyak 4 kelas)

y_i = label aktual untuk kelas ke-i (bernilai 1 untuk kelas benar dan 0 untuk lainnya)

p_i= probabilitas prediksi model untuk kelas ke-i

2.5.3 *Backpropagation*

Backpropagation merupakan algoritma fundamental dalam proses pelatihan model *Deep Learning* yang berfungsi untuk memperbarui parameter bobot (*weight*) secara iteratif guna meminimalkan nilai fungsi *loss*. Setelah proses aliran maju (*forward pass*) menghasilkan prediksi dan nilai kesalahan dihitung, algoritma ini bekerja dengan mengalirkan informasi kesalahan secara mundur dari lapisan keluaran (*output layer*) menuju lapisan masukan (*input layer*) [35]. Proses ini memanfaatkan aturan rantai (*chain rule*) dalam kalkulus untuk menghitung nilai gradien, yang merepresentasikan kontribusi setiap parameter terhadap total kesalahan prediksi. Nilai gradien memiliki peran penting karena menentukan arah serta besar langkah pembaruan parameter agar model dapat mencapai kondisi kesalahan minimum [36]. Pada kasus klasifikasi multi-kelas yang menggunakan fungsi aktivasi *Softmax* dan fungsi *loss Categorical Cross Entropy*, turunan parsial dari fungsi *loss* terhadap skor mentah (*logits*) dapat dirumuskan sebagai berikut:

$$\frac{\partial L_{CE}}{\partial z_i} = p_i - y_i \quad \text{..... (2.9)}$$

Dimana:

$\frac{\partial L_{CE}}{\partial z_i}$ = gradien fungsi *loss* terhadap logit ke-i.

p_i = probabilitas prediksi untuk kelas ke-i.

y_i = label aktual untuk kelas ke-i.

Setelah nilai gradien diperoleh pada setiap lapisan, dilakukan proses pembaruan parameter jaringan untuk mengurangi nilai kesalahan. Proses ini menggunakan algoritma optimasi, salah satunya adalah *Adam* (*Adaptive Moment Estimation*), yang mampu menyesuaikan nilai *learning rate* secara adaptif untuk setiap parameter. Algoritma ini mengombinasikan estimasi momen pertama, yaitu rata-rata gradien, dan momen kedua, yaitu rata-rata kuadrat gradien, sehingga menghasilkan proses konvergensi yang lebih cepat dan stabil [35]. Secara matematis, pembaruan parameter pada algoritma *Adam* dirumuskan sebagai berikut:

$$W_{baru} = W_{lama} - \alpha \times \frac{m_t}{\sqrt{v_t}+\epsilon} \quad \text{..... (2.10)}$$

Dimana:

W_{baru} = bobot setelah pembaruan.

W_{lama} = bobot sebelum pembaruan.

α = *learning rate*.

m_t = estimasi momen pertama (rata-rata gradien).

v_t = estimasi momen kedua (rata-rata kuadrat gradien).

ϵ = konstanta kecil untuk menghindari pembagian dengan nol.

2.5.4 Evaluasi Proses Pelatihan Model

Evaluasi proses pelatihan model dilakukan untuk memantau perkembangan kinerja model selama setiap *epoch* menggunakan metrik *loss* dan *accuracy*. Nilai *loss* merepresentasikan besarnya kesalahan prediksi yang dihasilkan model selama proses pembelajaran. Semakin rendah nilai *loss*, semakin baik kemampuan model dalam mempelajari pola yang terdapat pada data. Pada kondisi ideal, nilai *loss* akan terus menurun seiring bertambahnya *epoch*, yang menunjukkan bahwa model semakin mampu meminimalkan kesalahan prediksi. Sementara itu, metrik *accuracy* digunakan untuk mengukur tingkat ketepatan prediksi model dalam mengklasifikasikan data. Peningkatan nilai *accuracy* selama proses pelatihan mengindikasikan bahwa kemampuan model dalam mengenali pola semakin baik. Namun, setelah mencapai titik tertentu, perubahan nilai *loss* dan *accuracy* dapat menjadi relatif stabil atau memasuki kondisi datar (*plateau)*. Kondisi ini menunjukkan bahwa model telah mencapai batas peningkatan performa untuk konfigurasi pelatihan yang digunakan [37].

Agar evaluasi proses pelatihan lebih komprehensif, analisis tidak hanya dilakukan terhadap *training loss*, tetapi juga *validation loss*. *Training loss* menggambarkan kemampuan model dalam mempelajari pola dari data latih, sedangkan *validation loss* digunakan untuk mengevaluasi kemampuan model dalam melakukan prediksi terhadap data validasi yang tidak dilibatkan selama proses pelatihan. Perbandingan kedua metrik tersebut memberikan gambaran mengenai kemampuan generalisasi model, yaitu kemampuan model untuk mempertahankan performanya ketika dihadapkan pada data baru [37].

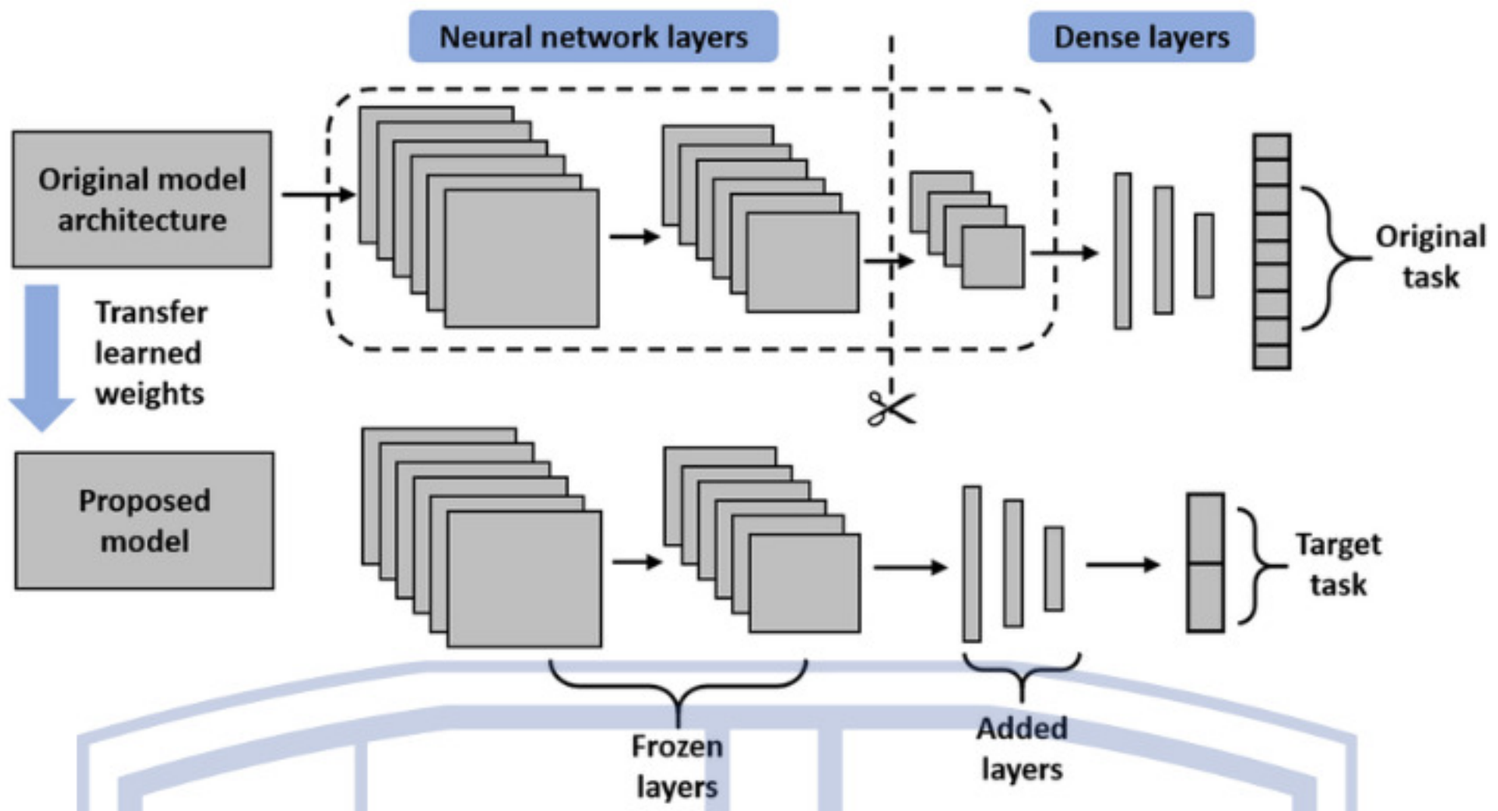
Selain menilai perkembangan performa model, analisis terhadap *training loss* dan *validation loss* juga bertujuan untuk mengidentifikasi potensi terjadinya *underfitting* maupun *overfitting*. Fenomena *overfitting* dapat disimpulkan terjadi apabila nilai fungsi *training loss* terus menurun seiring berjalannya waktu hingga menyentuh nilai rendah yang dapat diterima, namun nilai fungsi *validation loss* hanya menurun sampai titik tertentu lalu berbalik mengalami peningkatan. Di sisi lain, apabila nilai fungsi *loss* tetap berada di angka yang tinggi dan tidak menunjukkan penurunan seiring dengan bertambahnya jumlah iterasi baik pada kurva latih maupun validasi, hal ini mengindikasikan bahwa model yang

digunakan kurang kompleks dalam memetakan data, yang pada akhirnya berujung pada masalah *underfitting* [37].

2.6 *Transfer learning*

Pendekatan *transfer learning* dalam pembelajaran mendalam dikembangkan untuk mengatasi keterbatasan data pelatihan pada tugas tertentu dengan memanfaatkan model yang telah dilatih pada dataset berskala besar. Pada arsitektur *Convolutional Neural Network* (CNN), pendekatan ini memungkinkan penggunaan kembali representasi fitur yang telah dipelajari pada tahap sebelumnya untuk diterapkan pada tugas baru dengan karakteristik serupa. Pemanfaatan tersebut berimplikasi pada pengurangan kebutuhan pelatihan dari awal serta memungkinkan model tetap mampu mengekstraksi fitur yang relevan meskipun jumlah data pada dataset target terbatas [30].

Implementasi *transfer learning* umumnya dilakukan melalui penggunaan model pralatih (*pretrained model*) yang telah dilatih pada dataset besar, seperti *ImageNet*, kemudian disesuaikan dengan dataset target yang lebih spesifik. Representasi fitur yang dipelajari pada model pralatih mencakup pola dasar seperti tepi, tekstur, dan struktur visual, yang masih relevan untuk berbagai tugas dalam domain citra. Proses adaptasi dilakukan melalui dua pendekatan utama, yaitu *feature extraction* dan *fine-tuning*. Pada *feature extraction*, lapisan awal jaringan dipertahankan untuk mempertahankan fitur umum, sementara lapisan akhir dimodifikasi sesuai dengan jumlah kelas pada dataset target. Adapun pada *fine-tuning*, sebagian atau seluruh lapisan jaringan dilatih kembali untuk menyesuaikan representasi fitur dengan karakteristik data yang lebih spesifik. Kedua pendekatan ini memberikan fleksibilitas dalam menyesuaikan model terhadap kebutuhan tugas tanpa memerlukan pelatihan penuh dari awal [30], [38].



Gambar 2.7 *General transfer learning framework [39]*

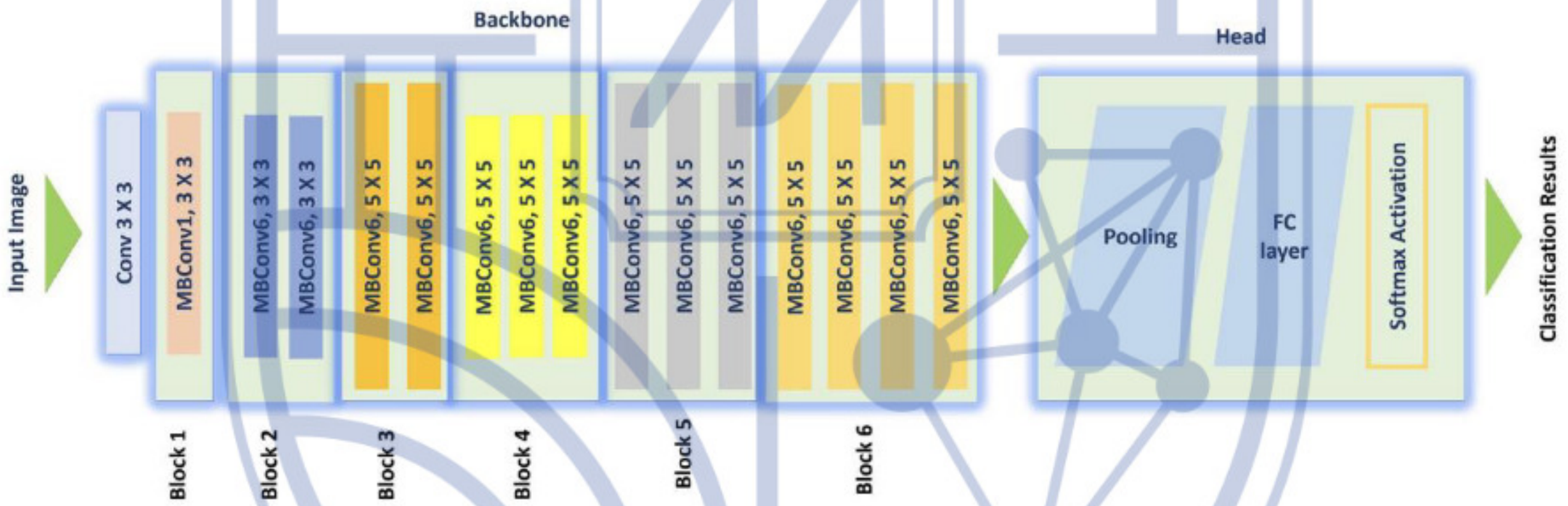
Penerapan *transfer learning* menunjukkan relevansi yang kuat pada permasalahan klasifikasi tingkat kematangan buah mangga, terutama pada kondisi ketersediaan data yang terbatas [40]. Pemanfaatan pengetahuan dari domain sumber memungkinkan model mencapai kinerja yang dapat diandalkan tanpa bergantung pada jumlah data pelatihan yang besar. Sejumlah penelitian melaporkan bahwa penggunaan model pralatih, seperti VGG16, mampu menghasilkan kinerja yang kompetitif dalam klasifikasi citra buah mangga, meskipun hasil yang diperoleh dipengaruhi oleh karakteristik dataset yang digunakan [6], [40]. Berdasarkan pertimbangan tersebut, penelitian ini mengadopsi arsitektur EfficientNet-B0 yang dirancang untuk meningkatkan efisiensi komputasi melalui mekanisme penskalaan jaringan yang terstruktur, sehingga mendukung proses klasifikasi dengan kompleksitas yang lebih terkendali.

2.7 EfficientNet-B0

EfficientNet-B0 diperkenalkan sebagai model dasar dalam arsitektur *Convolutional Neural Network* (CNN) yang mengadopsi metode Compound Scaling untuk menyeimbangkan dimensi kedalaman (*depth*), lebar (*width*), dan resolusi (*resolution*) jaringan secara proporsional. Berbeda dengan pendekatan pada arsitektur CNN konvensional yang biasanya menyesuaikan ketiga dimensi tersebut secara terpisah, metode *Compound Scaling* dirancang untuk menjaga efisiensi model secara keseluruhan. Ketiga dimensi tersebut memiliki keterkaitan yang erat; misalnya, peningkatan resolusi citra input akan menuntut penggunaan filter dengan ukuran yang lebih besar agar proses konvolusi tetap berjalan optimal [41]. Dengan menerapkan rasio penskalaan yang konsisten dan

tersinkronisasi, model dapat meningkatkan kapasitasnya tanpa mengorbankan efisiensi komputasi [41].

Penentuan nilai koefisien penskalaan pada *compound scaling* dilakukan melalui pencarian arsitektur guna menemukan kombinasi yang menghasilkan tingkat akurasi optimal [41]. Pendekatan ini dieksekusi secara langsung pada model *baseline* dasar karena pencarian pada model berukuran besar membutuhkan biaya komputasi yang jauh lebih tinggi. Oleh karena itu, koefisien yang diperoleh dari *baseline* tersebut kemudian diterapkan untuk seluruh varian yang lebih besar [11]. Dalam kerangka ini, EfficientNet-B0 ditetapkan sebagai arsitektur *baseline* utama tanpa adanya penskalaan tambahan pada dimensi kedalaman, lebar, maupun resolusinya. Kondisi ini menjadikan EfficientNet-B0 sebagai varian titik awal dengan jumlah parameter, *Floating-Point Operations* (FLOPs), dan kebutuhan komputasi paling kecil di antara keluarga EfficientNet, sehingga sangat relevan digunakan pada penelitian yang mengutamakan efisiensi namun tetap menuntut performa prediksi yang akurat [11], [41].



Gambar 2.8 *Architecture of EfficientNet-B0 with MBConv as basic building Blocks* [13]

Pembahasan komprehensif mengenai komponen-komponen arsitektur EfficientNet-B0 pada Gambar 2.8 dapat diuraikan sebagai berikut:

1. *Input layer*

   Lapisan masukan bertugas menerima citra input dan meneruskannya ke dalam jaringan model. Pada tahap ini, citra umumnya diubah ukurannya (*resize*) menjadi dimensi 224×224 piksel, atau pada beberapa kasus yang lebih jarang, menjadi 260×260 piksel [13].

2. *Stem Convolution*

   Setiap citra masukan akan melalui serangkaian prosedur konvolusi (ditunjukkan oleh blok Conv 3×3 pada awal *Backbone*) untuk mengekstraksi fitur-fitur tingkat rendah

(*low-level features*). Umumnya, citra masukan tersebut diproses melalui serangkaian operasi konvolusi berukuran 3×3 [13].

3. *EfficientNet Blocks*

Model ini terdiri dari beberapa blok yang di dalamnya mencakup serangkaian lapisan *Mobile Inverted bottleneck Convolution* (MBConv). Lapisan MBConv merupakan komponen fundamental dari EfficientNet yang mengintegrasikan inverted residual connections dengan teknik *4*. Setiap blok MBConv bertugas untuk mengekstraksi serta menyempurnakan fitur pada berbagai tingkat abstraksi [13]. Arsitektur EfficientNet-B0 tersusun atas blok-blok MBConv yang diaplikasikan secara berulang dan berjenjang (*stages*). Setiap tingkatan memiliki kedalaman dan lebar yang berbeda-beda, dimulai dari lapisan konvolusi awal hingga blok MBConv6 [42]. Pada setiap blok (diilustrasikan dari *Block* 1 hingga *Block* 6), digunakan operasi konvolusi dengan kernel berukuran 3×3 dan 5×5. Seiring bertambahnya kedalaman jaringan, dimensi resolusi citra mengalami reduksi spasial secara bertahap, sementara jumlah kanal meningkat secara progresif guna meningkatkan kemampuan model dalam mengenali pola fitur yang lebih kompleks. Mekanisme hierarkis ini menghasilkan representasi akhir berupa *feature map* berukuran 7×7×1280 sekaligus mempertahankan efisiensi komputasi [41]. Rincian spesifikasi dari setiap tingkatan (*stages*) tersebut, yang mencakup jenis operator, ukuran resolusi spasial, hingga ekspansi jumlah kanal yang digunakan, dirangkum secara komprehensif pada Tabel 2.1.

Tabel 2.1 EfficientNet-B0 *Architecture* [41]

Stage i	Operator $\hat{F}_i$	Resolution $\hat{H}_i X \hat{W}_i$	Channels $\hat{C}_i$	Layers $\hat{L}_i$
1	Conv3x3	224 x 224	32	1
2	MBConv1, k3x3	112 x 112	16	1
3	MBConv6, k3x3	112 x 112	24	2
4	MBConv6, k5x5	56 x56	40	2
5	MBConv6, k3x3	28 x 28	80	3
6	MBConv6, k5x5	14 x 14	112	3
7	MBConv6, k5x5	14 x 14	192	4
8	MBConv6, k3x3	7 x 7	320	1
9	Conv & *Pooling* & FC	7 x 7	1280	1

Berdasarkan Tabel 2.1, arsitektur EfficientNet-B0 menerima citra masukan berukuran 224 × 224 piksel dengan tiga kanal warna pada *stage* 1. Citra tersebut kemudian

mengalami reduksi dimensi spasial secara progresif melalui sembilan *stage*, dengan setiap baris pada tabel menampilkan ukuran resolusi (H × W), jumlah kanal (C), serta jumlah *layer* (L) penyusun *stage* tersebut. Resolusi spasial pada masing-masing *stage* menyusut secara bertahap dari 224 × 224 menjadi 112 × 112, 56 × 56, 28 × 28, 14 × 14, hingga mencapai 7 × 7 pada *stage* 8 dan *stage* 9. Bersamaan dengan reduksi spasial tersebut, jumlah kanal meningkat secara progresif dari 32 pada *stage* 1 menjadi 1280 pada *stage* 9, yang merepresentasikan peningkatan kapasitas model dalam menangkap fitur visual dengan tingkat abstraksi yang semakin tinggi. Pada bagian akhir arsitektur, operasi konvolusi 1 × 1 yang dikombinasikan dengan *pooling* dan *fully connected layer* menghasilkan *feature map* akhir berukuran 7 × 7 × 1280 yang diteruskan ke lapisan klasifikasi. Mekanisme reduksi spasial dan ekspansi kanal yang berjenjang inilah yang menjadi dasar bagi EfficientNet-B0 untuk mengekstraksi fitur secara hierarkis sekaligus mempertahankan efisiensi komputasi [41]. Secara struktural, MBConv dirancang untuk mencapai keseimbangan antara akurasi dan efisiensi penggunaan parameter. Berbeda dengan konvolusi konvensional, MBConv menerapkan struktur *inverted bottleneck*, yaitu memperluas jumlah saluran (*channel expansion*) menggunakan konvolusi sebelum proses ekstraksi fitur dilakukan. Setelah tahap ekspansi, fitur diproses menggunakan *Depthwise Separable Convolution* yang terdiri atas *Depthwise Convolution* dan *Pointwise Convolution* untuk mengurangi kompleksitas komputasi tanpa mengorbankan kemampuan representasi fitur [43]. Setelah diekspansi, ekstraksi fitur dilakukan menggunakan *Depthwise Separable Convolution* yang memecah proses menjadi *Depthwise Convolution* dan *Pointwise Convolution* untuk menjaga efisiensi. Untuk menjaga stabilitas proses pelatihan, keluaran dari setiap operasi konvolusi dinormalisasi menggunakan *Batch Normalization* (BN), yang menyesuaikan nilai rata-rata dan varians aktivasi pada setiap *mini-batch* sehingga mempercepat konvergensi dan meningkatkan kemampuan generalisasi model [13]. Selanjutnya, hasil ekstraksi fitur diproyeksikan kembali ke jumlah saluran yang lebih kecil dan dikombinasikan dengan masukan awal melalui *residual connection* guna mempertahankan aliran informasi dan mempermudah proses optimasi jaringan [43]. Selain itu, modul *Squeeze-and-Excitation (SE) Block* diintegrasikan ke dalam MBConv sebagai mekanisme perhatian (*channel attention*) yang memberikan bobot adaptif pada setiap saluran fitur sehingga model dapat lebih fokus pada informasi visual yang relevan [44].

4. *Global Average Pooling layer (Pooling)*

Mengikuti lapisan-lapisan konvolusi pada *Backbone*, lapisan GAP (ditunjukkan sebagai blok *Pooling* pada bagian *Head*) digunakan untuk menghitung nilai rata-rata dari setiap *feature map*. Proses ini secara esensial memadatkan informasi spasial menjadi sebuah vektor tunggal [13].

5. *Fully connected dan Output layers*

   Setelah melewati lapisan GAP, biasanya terdapat satu atau beberapa lapisan *fully connected* (*FC layer*). Tujuan dari lapisan ini adalah untuk mentransformasikan fitur-fitur tingkat tinggi yang diperoleh dari lapisan konvolusi menjadi kelas-kelas *output* yang dibutuhkan[13] . Selanjutnya, *output layer* bertugas menghasilkan probabilitas klasifikasi akhir untuk citra masukan yang diberikan. Lapisan terakhir ini umumnya memiliki fungsi aktivasi *Softmax* (*Softmax Activation*), yang mengubah keluaran model yang belum diproses menjadi nilai probabilitas untuk setiap kelas guna menghasilkan *Classification Results* [13].

Dalam penerapannya, arsitektur dasar model ini diadaptasi untuk kebutuhan spesifik klasifikasi melalui modifikasi pada lapisan-lapisan terakhir di bagian *Head*. Fitur-fitur spasial yang telah dipadatkan oleh lapisan *Global Average Pooling* diteruskan ke *Dense layer* menggunakan fungsi aktivasi ReLU untuk mempelajari hubungan nonlinier antarfitur sebelum menghasilkan probabilitas kelas melalui fungsi aktivasi Softmax pada lapisan keluaran [13]. Untuk mengurangi risiko *overfitting*, setelah *Dense layer* disisipkan lapisan *Dropout* yang menonaktifkan sebagian neuron secara acak selama proses pelatihan. Mekanisme ini mencegah model terlalu bergantung pada neuron tertentu sehingga representasi fitur yang dipelajari menjadi lebih robust dan kemampuan generalisasi model dapat ditingkatkan [45].

2.8 Confusion Matrix

Confusion Matrix merupakan alat fundamental dalam mengevaluasi kinerja model klasifikasi, yang menyajikan ringkasan terstruktur mengenai perbandingan antara hasil prediksi model dengan kelas aktualnya [46]. Keempat nilai di dalam matriks ini menjadi dasar dalam memperoleh berbagai metrik evaluasi yang menggambarkan performa model pada tugas klasifikasi.

Tabel 2.2 Confusion Matrix

Nilai Prediksi	Nilai Aktual	
	Positif	Negatif

Positif	TP	FN
Negatif	FP	TN

Akurasi mengukur proporsi data yang diklasifikasikan dengan benar dari keseluruhan data uji [46]. Akurasi dihitung dengan rumus berikut.

$$Accuracy \frac{TP+TN}{TP+FP+FN+TN} \quad \text{..... (2.11)}$$

Precision adalah proporsi prediksi benar pada kelas positif dibandingkan dengan keseluruhan prediksi positif [46]. Berikut ini persamaan untuk menghitung nilai *Precision*.

$$Precision = \frac{TP}{TP+FP} \quad \text{..... (2.12)}$$

Recall adalah proporsi prediksi benar pada kelas positif dibandingkan dengan total data aktual pada kelas tersebut [46]. Nilai *Recall* dihitung menggunakan rumus berikut.

$$Recall = \frac{TP}{TP+FN} \quad \text{..... (2.13)}$$

F1-Score merupakan rata-rata harmonis dari Precision dan Recall [46]. Berikut ini persamaan untuk menghitung nilai *F1-Score*.

$$F1\ Score = \frac{2 * Precision * Recall}{Precision + Recall} \quad \text{..... (2.14)}$$

2.9 Evaluasi Efisiensi Komputasi Model

Evaluasi efisiensi komputasi model dilakukan untuk menilai kebutuhan sumber daya selama proses operasional model, terlepas dari hasil klasifikasi yang dicapai. Penilaian ini menjadi pertimbangan penting ketika model akan diterapkan pada lingkungan dengan keterbatasan perangkat keras, sehingga pengembangan model *Deep Learning* umumnya difokuskan pada tiga aspek, yaitu pengurangan kebutuhan memori, penurunan jumlah operasi *Floating-Point*, dan pemeliharaan tingkat akurasi [13]. Efisiensi tersebut diukur melalui empat indikator yang dapat dikelompokkan ke dalam dua dimensi, yaitu kompleksitas struktural model yang mencakup jumlah parameter dan ukuran model, serta kompleksitas komputasi yang mencakup jumlah FLOPs dan waktu inferensi. Kedua dimensi tersebut saling berkaitan, di mana jumlah parameter dan FLOPs pada arsitektur CNN cenderung berkembang secara beriringan dengan koefisien korelasi sebesar 0,772 [47]. Meskipun demikian, keempat indikator tetap diukur secara bersamaan karena masing-masing merepresentasikan aspek efisiensi yang berbeda dan tidak sepenuhnya saling menggantikan. Keempat indikator tersebut diuraikan sebagai berikut.

1. Jumlah parameter model

   Menunjukkan banyaknya bobot yang dimiliki model. Semakin besar jumlah parameter, semakin tinggi kompleksitas model serta kebutuhan penyimpanan yang diperlukan.

2. Ukuran model (*model size*)

   Menggambarkan kapasitas memori yang dibutuhkan untuk menyimpan parameter model, umumnya dinyatakan dalam satuan *megabyte* (MB). Ukuran model dapat dihitung berdasarkan jumlah parameter yang digunakan, dengan asumsi setiap parameter direpresentasikan dalam 32-bit (4 *byte*).

3. Jumlah operasi *Floating-Point* (FLOPs)

   Menunjukkan jumlah operasi aritmatika yang dilakukan model selama proses inferensi. Nilai ini digunakan untuk merepresentasikan tingkat kompleksitas komputasi yang dibutuhkan oleh model.

4. Waktu inferensi

   Menggambarkan waktu yang diperlukan model untuk menghasilkan keluaran dari satu data masukan. Parameter ini digunakan untuk menilai efisiensi waktu dalam proses prediksi.

Interpretasi terhadap nilai keempat indikator memerlukan standar acuan agar penilaian efisiensi bersifat objektif dan dapat dipertanggungjawabkan. Pada arsitektur EfficientNet-B0, jumlah parameter tercatat sebesar 5,3 juta dengan kebutuhan komputasi 0,39 miliar FLOPs, yang tergolong rendah dibandingkan arsitektur CNN konvensional [13]. Perbandingan pada penelitian [13] menunjukkan EfficientNet-B0 hanya membutuhkan memori sebesar 15,64 MB, sedangkan VGG16 membutuhkan 56,21 MB, Inception-v3 sebesar 83,48 MB, dan ResNet-50 sebesar 90,28 MB, sehingga EfficientNet-B0 menempati profil kebutuhan memori paling rendah di antara model pembanding tersebut. Profil kebutuhan sumber daya yang rendah ini menjadikan EfficientNet-B0 sesuai untuk penerapan pada sistem dengan keterbatasan perangkat keras [13]. Nilai-nilai acuan tersebut digunakan sebagai dasar untuk menilai apakah suatu model tergolong efisien atau tidak efisien pada penelitian ini.

Selain mengacu pada standar dari literatur, evaluasi efisiensi pada penelitian ini dilakukan melalui perbandingan langsung dengan arsitektur CNN ringan lain, yaitu MobileNetV2. Arsitektur tersebut dipilih sebagai pembanding karena memiliki orientasi desain yang serupa, yaitu model ringan untuk klasifikasi citra dengan jumlah parameter sekitar 3,5 juta [48]. MobileNetV2 juga telah digunakan dalam penelitian klasifikasi tingkat kematangan buah, sehingga relevan dijadikan acuan pembanding dalam domain penelitian

ini [49]. Kedua arsitektur dievaluasi menggunakan dataset yang sama agar perbandingan keempat indikator efisiensi dilakukan secara setara. Dengan demikian, penilaian efisiensi model tidak hanya mengacu pada literatur, tetapi juga pada hasil perbandingan langsung dengan MobileNetV2 yang dibahas lebih lanjut pada Bab IV dan Bab V.

2.10 Tingkat Kematangan Buah Mangga

Tingkat kematangan buah mangga merupakan indikator yang digunakan untuk menilai kualitas buah serta menentukan waktu panen yang sesuai. Pada komoditas hortikultura seperti mangga (*Mangifera indica*), proses pematangan berkaitan dengan perubahan karakteristik fisik dan kimia yang meliputi warna kulit, tekstur, rasa, serta aroma buah. Perubahan tersebut terjadi sebagai bagian dari proses fisiologis selama pematangan yang menyebabkan warna kulit mangga berubah secara bertahap dari hijau menuju kuning atau kemerahan seiring meningkatnya tingkat kematangan [50]. Kondisi kematangan buah juga berhubungan dengan tingkat penerimaan konsumen dan nilai jual produk di pasar, sehingga identifikasi tingkat kematangan menjadi aspek penting dalam kegiatan produksi dan distribusi buah. Penentuan tingkat kematangan yang tepat memungkinkan penetapan waktu panen yang sesuai serta menjaga kualitas buah selama proses penanganan pascapanen [51]

Identifikasi tingkat kematangan buah mangga umumnya dilakukan melalui pengamatan terhadap sejumlah parameter fisik dan sensorik yang dapat diamati secara langsung. Parameter tersebut meliputi perubahan warna kulit, tekstur buah, aroma, serta tingkat kemanisan yang berkaitan dengan kandungan gula pada buah [52]. Selama proses pematangan, warna kulit mangga mengalami perubahan bertahap dari hijau menuju kuning atau kekuningan, yang disertai perubahan tekstur dari keras menjadi lebih lunak serta munculnya aroma khas buah matang [50]. Meskipun demikian, metode penilaian yang dilakukan melalui pengamatan visual secara langsung oleh manusia cenderung bersifat subjektif dan dapat menghasilkan variasi penilaian antar pengamat [53]. Kondisi tersebut mendorong pemanfaatan pendekatan yang lebih terukur melalui teknologi pengolahan citra digital untuk mengidentifikasi karakteristik visual buah secara sistematis.

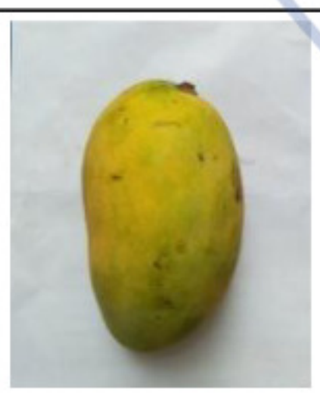
Perkembangan metode pengolahan citra digital memungkinkan tingkat kematangan buah diperlakukan sebagai objek klasifikasi dalam sistem komputer. Dalam pendekatan ini, citra buah digunakan sebagai data masukan yang mengandung informasi visual seperti warna, bentuk, dan tekstur yang dapat dijadikan fitur untuk membedakan setiap tingkat kematangan. Teknik *computer vision* dan *deep learning*, khususnya model *Convolutional Neural Network* (CNN), banyak digunakan untuk mengekstraksi fitur visual dari citra secara

otomatis sehingga memungkinkan proses klasifikasi tingkat kematangan buah dilakukan secara komputasional. Pendekatan tersebut telah digunakan dalam berbagai penelitian pengolahan citra untuk mendeteksi perubahan visual pada buah yang berkaitan dengan tingkat kematangan [7].

Dalam penelitian ini, tingkat kematangan buah mangga diklasifikasikan ke dalam empat kategori, yaitu *unripe* (belum matang), *partially ripe* (setengah matang), *ripe* (matang konsumsi), dan *rotten* (busuk). Setiap kategori memiliki karakteristik visual yang berbeda yang dapat diamati dari perubahan warna kulit serta kondisi fisik buah. Perbedaan karakteristik tersebut digunakan sebagai dasar dalam proses pelabelan data citra yang digunakan untuk melatih model klasifikasi.

Tabel 2.3 berikut menunjukkan kategori tingkat kematangan buah mangga yang digunakan dalam penelitian ini beserta deskripsinya.

Tabel 2.3 Kategori Tingkat Kematangan Buah Mangga pada Penelitian [7]

Contoh Citra Mangga	Kategori	Deskripsi
	*Unripe* (Mentah)	Buah mangga dengan warna kulit hijau tua, tekstur masih keras, dan belum memiliki aroma khas mangga matang. Pada kondisi ini buah belum siap untuk dikonsumsi.
	*Partially ripe* (Setengah Matang)	Mangga mulai mengalami perubahan warna dari hijau menjadi kekuningan, namun warna hijau masih cukup dominan. Tekstur buah mulai sedikit melunak tetapi belum sepenuhnya matang.
	*Ripe* (Matang)	Mangga memiliki warna kuning atau kuning kehijauan dengan tekstur yang lebih lunak dan aroma khas buah matang. Pada tahap ini buah berada pada kondisi optimal untuk dikonsumsi.
	*Rotten* (Busuk)	Mangga mengalami penurunan mutu yang ditandai dengan warna cokelat kehitaman, tekstur sangat lunak atau lembek, serta munculnya tanda-tanda kerusakan pada permukaan buah sehingga tidak layak dikonsumsi.