BAB 2
LANDASAN TEORI

2.1 Penelitian Terdahulu

Penerapan *Computer Vision* dalam bidang dermatologi telah membantu proses identifikasi kondisi kulit secara lebih objektif [3]. Akan tetapi, model deteksi objek seperti YOLO masih dapat mengalami keterbatasan ketika kelas objek memiliki karakteristik visual yang mirip atau ketika hasil deteksi perlu diklasifikasikan lebih lanjut. Pendekatan *Hybrid Machine Learning* menjadi relevan karena detektor dapat digunakan untuk menemukan lokasi objek, sedangkan algoritma klasifikasi lanjutan dapat digunakan untuk menganalisis area objek yang telah terdeteksi. Ringkasan penelitian terdahulu yang digunakan untuk menjelaskan posisi dan gap penelitian ditunjukkan pada Tabel 2.1.

Tabel 2.1. Perbandingan dan gap penelitian

No	Peneliti (Tahun)	Metode dan Algoritma	Objek dan Dataset	Hasil dan Keterbatasan
1	Luu dkk. (2025) [8]	YOLOv10, *Random Forest*, dan PCA	Enam jenis sampah. *Dataset Trash Classification* (5.771 citra).	Akurasi YOLOv10 murni 90,04%, YOLOv10+*Random Forest* 90,41%, dan konfigurasi dengan PCA 91,14%; terdapat potensi atenuasi detail visual akibat PCA serta peningkatan latensi.
2	Ulfa dkk. (2026) [9]	YOLOv8n, seleksi fitur berbasis *Random Forest*, SMOTE, *Logistic Regression*, SVM, dan *XGBoost*	Lima tingkat kematangan alpukat. *Dataset* alpukat lokal Aceh (736 citra).	Akurasi YOLOv8n murni 90% dan model *hybrid* 96%; produk berupa aplikasi Android yang memerlukan instalasi serta performa bergantung pada spesifikasi perangkat keras.
3	Harahap dkk. (2025) [10]	YOLO dan *Support Vector Machine*	Kualitas buah jeruk lokal. *Dataset* berukuran kecil (120 citra).	Akurasi model *hybrid* mencapai 91,6%; jumlah data terbatas dan performa deteksi dipengaruhi stabilitas pencahayaan pada lingkungan terkontrol.
4	Maitimu dkk. (2025) [11]	YOLOv8, *K-Nearest Neighbor*, dan *Random Forest*	Tingkat keparahan jerawat. Citra wajah berjerawat.	Akurasi *hybrid* KNN 95% dan *hybrid* RF 97%; klasifikasi terbatas pada tingkat keparahan umum tanpa identifikasi jenis lesi jerawat secara spesifik.
5	Penelitian ini	YOLO26, *Random Forest* berbasis fitur LAB, dan *confidence-based routing*	Empat jenis lesi jerawat: Blackheads, Papules, Pustules, dan Whiteheads. *Dataset Acne Detection V2* dan *acnee*.	Menggunakan YOLO26 sebagai detektor dan *Random Forest* sebagai klasifikasi lanjutan pada *crop* lesi berbasis fitur LAB 63 dimensi.

Penelitian Luu dkk. menunjukkan bahwa *Random Forest* dapat digunakan sebagai komponen lanjutan setelah YOLO pada klasifikasi enam jenis sampah. Akurasi meningkat dari 90,04% pada YOLOv10 murni menjadi 90,41% pada kombinasi YOLOv10 dan *Random Forest*, serta 91,14% pada konfigurasi yang melibatkan PCA. Penelitian ini menjadi salah satu dasar bahwa hasil deteksi YOLO dapat dibantu oleh algoritma klasifikasi eksternal. Namun, objek penelitian tersebut bukan citra kulit dan belum membahas karakteristik visual lesi jerawat.

Ulfa dkk. memadukan YOLOv8n, seleksi fitur berbasis *Random Forest*, SMOTE, *Logistic Regression*, *Support Vector Machine*, dan *XGBoost* untuk klasifikasi lima tingkat kematangan alpukat. Harahap dkk. menggunakan kombinasi YOLO dan *Support Vector Machine* untuk klasifikasi kualitas jeruk lokal. Kedua penelitian tersebut menunjukkan bahwa pendekatan *hybrid* dapat digunakan pada klasifikasi objek berbasis citra. Akan tetapi, keduanya berada pada domain objek umum atau buah, sehingga belum menjawab kebutuhan klasifikasi lesi jerawat.

Pada domain jerawat, Maitimu dkk. menggabungkan YOLOv8, *K-Nearest Neighbor*, dan *Random Forest* untuk klasifikasi tingkat keparahan jerawat. Penelitian tersebut memperoleh akurasi 95% pada KNN dan 97% pada *Random Forest*. Meskipun sudah berada pada domain jerawat, fokus klasifikasinya masih berada pada tingkat keparahan umum. Dengan demikian, penelitian tersebut belum membahas klasifikasi jenis lesi jerawat secara spesifik.

Berdasarkan tinjauan tersebut, gap penelitian ini terletak pada penggunaan *Random Forest* sebagai klasifikasi lanjutan untuk empat jenis lesi jerawat pada *crop* hasil deteksi YOLO26. Penelitian ini tidak hanya memanfaatkan YOLO26 untuk mendeteksi lokasi lesi, tetapi juga mengekstraksi fitur LAB 63 dimensi dari area lesi untuk diproses oleh *Random Forest*. Kelas yang dianalisis adalah *Blackheads*, *Papules*, *Pustules*, dan *Whiteheads*. Evaluasi dilakukan pada dua skenario pembagian *dataset* dan difokuskan pada metrik klasifikasi multi-kelas, pola kesalahan per kelas, serta pengujian citra nyata.

2.2 Jerawat (Acne Vulgaris)

Jerawat atau *acne vulgaris* merupakan kondisi inflamasi kronis pada unit folikel rambut dan kelenjar sebasea. Kondisi ini dapat muncul sebagai komedo, papula, pustula, kista, dan nodul, terutama pada area kulit yang kaya kelenjar sebasea seperti wajah. Pembentukan jerawat dipengaruhi oleh hiperplasia kelenjar sebasea, peningkatan produksi sebum, keratinisasi abnormal pada saluran folikel, kolonisasi mikroba, serta respons inflamasi pada kulit [14]. Dalam analisis citra, lesi jerawat dapat diamati melalui karakteristik visual seperti warna, bentuk, kontras, serta keberadaan area terang atau gelap pada lesi. Ruang warna LAB relevan digunakan karena mampu memisahkan komponen kecerahan dan warna, sehingga dapat membantu analisis perbedaan visual antar lesi [15].

Penelitian ini berfokus pada empat kelas lesi jerawat yang menjadi target klasifikasi sistem, yaitu *Blackheads*, *Papules*, *Pustules*, dan *Whiteheads*. Keempat kelas tersebut dipilih karena mewakili dua kelompok lesi jerawat yang dapat diamati secara visual. *Blackheads* dan *Whiteheads* termasuk lesi noninflamasi, sedangkan *Papules* dan *Pustules* termasuk lesi inflamasi. Perbedaan kelompok tersebut berkaitan dengan ciri visual seperti titik gelap, area putih atau pucat, kemerahan, serta keberadaan pusat lesi yang lebih terang [14]. Selain tersedia pada *dataset*, karakter visual keempat kelas tersebut relevan untuk dianalisis melalui fitur warna, kecerahan, dan kontras pada ruang warna LAB [12, 15].

1. **Blackheads (Komedo Terbuka)**
   Blackheads merupakan lesi noninflamasi yang berkaitan dengan penumpukan sebum dan keratinosit di dalam folikel. Warna gelap pada *Blackheads* terbentuk akibat oksidasi sebum yang bercampur dengan debris pada permukaan kulit, sehingga lesi tampak sebagai titik gelap yang kontras terhadap kulit di sekitarnya [14]. Contoh karakteristik visual *Blackheads* ditunjukkan pada Gambar 2.1.

Gambar 2.1. Karakteristik visual blackheads
Sumber: Dataset Kaggle oleh Tiswan [16]

2. **Whiteheads (Komedo Tertutup)**

   Whiteheads merupakan lesi noninflamasi yang terbentuk akibat akumulasi sebum dan keratinosit di dalam folikel. Secara visual, lesi ini tampak sebagai tonjolan kecil berwarna putih atau pucat. Pada citra wajah, karakteristik tersebut dapat membuat *Whiteheads* memiliki kontras rendah terhadap kulit di sekitarnya [14]. Contoh karakteristik visual *Whiteheads* ditunjukkan pada Gambar 2.2.

Gambar 2.2. Karakteristik visual whiteheads
Sumber: Dataset Kaggle oleh Tiswan [16]

3. **Papules (Papula)**

   Papules merupakan salah satu bentuk lesi inflamasi pada jerawat. Secara visual, kelas ini tampak sebagai benjolan kecil kemerahan tanpa titik nanah di bagian tengah. Warna kemerahan pada *Papules* menunjukkan adanya proses

inflamasi, sehingga komponen warna yang merepresentasikan area merah menjadi penting dalam proses analisis citra [14, 15]. Contoh karakteristik visual *Papules* ditunjukkan pada Gambar 2.3.

Gambar 2.3. Karakteristik visual papules
Sumber: Dataset Kaggle oleh Tiswan [16]

4. **Pustules (Pustula)**

   Pustules merupakan salah satu bentuk lesi inflamasi pada jerawat yang memiliki kemiripan visual dengan *Papules*, tetapi disertai area pusat yang tampak lebih terang akibat akumulasi nanah. Secara visual, kelas ini memiliki kombinasi area kemerahan di sekitar lesi dan area putih atau kekuningan pada bagian pusat. Kemiripan visual dengan *Papules* dan *Whiteheads* membuat *Pustules* menjadi salah satu kelas yang menantang untuk diklasifikasikan [14, 15]. Contoh karakteristik visual *Pustules* ditunjukkan pada Gambar 2.4.

Gambar 2.4. Karakteristik visual pustules
Sumber: Dataset Kaggle oleh Tiswan [16]

2.3 Tipe Kulit Fitzpatrick

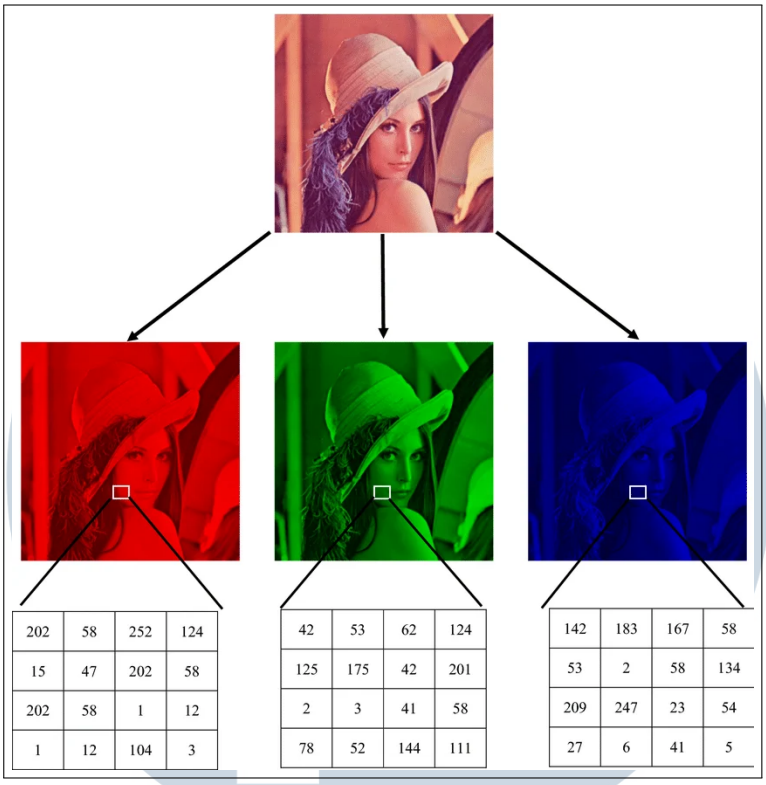
Tipe kulit Fitzpatrick merupakan klasifikasi karakteristik kulit yang berkaitan dengan respons kulit terhadap paparan sinar ultraviolet, terutama kecenderungan kulit mengalami kemerahan atau penggelapan. Dalam konteks analisis citra dermatologi, variasi tipe kulit perlu diperhatikan karena warna dasar kulit dapat memengaruhi kontras antara lesi dan area kulit di sekitarnya.

Penelitian Jurairattanaporn dkk. menunjukkan bahwa studi dermatoskopi pada pasien Asia dengan hiperpigmentasi pascainflamasi terkait jerawat melibatkan subjek dengan tipe kulit Fitzpatrick III sebesar 66% dan tipe IV sebesar 34%. Temuan tersebut relevan dengan penelitian ini karena citra yang digunakan berada pada rentang tipe kulit Fitzpatrick III dan IV. Oleh karena itu, interpretasi performa sistem dibatasi pada karakteristik citra kulit yang sesuai dengan rentang tipe kulit tersebut [17].

2.4 Computer Vision dan Machine Learning

Computer Vision merupakan bidang ilmu yang memungkinkan komputer menganalisis informasi visual dari citra atau video. Dalam penelitian ini, *Computer Vision* digunakan untuk mendeteksi lokasi lesi jerawat pada citra wajah, memotong area lesi, dan menyiapkan citra tersebut untuk proses klasifikasi. Citra digital direpresentasikan sebagai matriks numerik multidimensi yang menyimpan nilai intensitas warna pada setiap piksel [18]. Pada citra berwarna, setiap piksel umumnya memiliki tiga komponen warna, yaitu merah, hijau, dan biru pada ruang warna RGB. Setiap komponen tersebut disimpan dalam bentuk angka, sehingga pola visual pada citra dapat diproses oleh algoritma komputer.

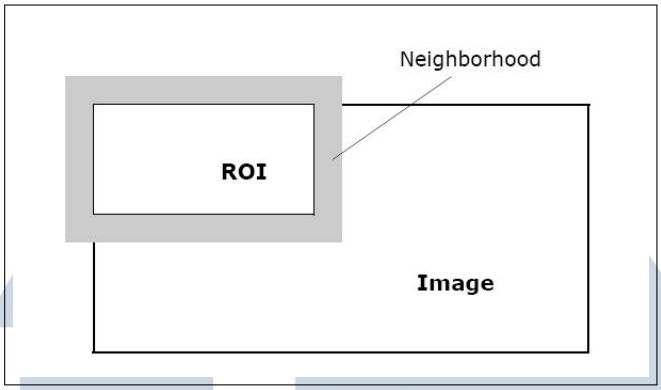
Representasi numerik tersebut menjadi dasar bagi proses deteksi dan klasifikasi berbasis citra. Model deteksi seperti YOLO26 memanfaatkan informasi piksel untuk mengenali lokasi objek dan menghasilkan *bounding box*. Sementara itu, pada penelitian ini, *Random Forest* tidak menerima citra mentah secara langsung, melainkan menerima vektor fitur hasil ekstraksi dari potongan citra lesi. Dengan demikian, citra wajah terlebih dahulu diubah menjadi bentuk numerik yang dapat dianalisis, baik oleh model deteksi maupun model klasifikasi lanjutan. Ilustrasi representasi citra digital dalam saluran RGB ditunjukkan pada Gambar 2.5.



Gambar 2.5. Representasi matriks numerik citra digital dalam saluran RGB
Sumber: Artikel jurnal Es Sabry dkk. [19]

Machine Learning merupakan pendekatan komputasi yang memungkinkan model mempelajari pola dari data. Pada klasifikasi citra, model *Machine Learning* membutuhkan representasi numerik yang mampu menggambarkan karakteristik objek. Berbeda dengan model *Deep Learning* yang dapat mempelajari fitur langsung dari citra dalam jumlah besar, algoritma seperti *Random Forest* lebih sesuai digunakan bersama fitur terstruktur yang telah diekstraksi sebelumnya [18, 20].

Dalam sistem ini, YOLO26 digunakan untuk mendeteksi area lesi jerawat dan menghasilkan *bounding box*. Area di dalam *bounding box* tersebut digunakan sebagai *Region of Interest* (ROI) untuk proses ekstraksi fitur dan klasifikasi lanjutan. Pemisahan ROI bertujuan agar *classifier* dapat berfokus pada area lesi, bukan pada keseluruhan wajah yang mengandung banyak informasi di luar objek target [18, 21]. Ilustrasi konsep ROI pada pemrosesan citra ditunjukkan pada Gambar 2.6.



Gambar 2.6. Ilustrasi konsep region of interest pada pemrosesan citra
Sumber: Dokumentasi web Intel [21]

2.5 Ruang Warna CIELAB

Ruang warna CIELAB atau CIE L*a*b* merupakan *color space* yang dikembangkan oleh CIE, yaitu organisasi internasional yang menetapkan standar pengukuran warna. Secara teknis, CIE L*a*b* merupakan pemetaan non-linear dari ruang warna XYZ agar perbedaan luminansi dan krominansi dapat direpresentasikan lebih seragam secara perseptual [18]. Pada penulisan CIE L*a*b*, tanda bintang (*) bukan menunjukkan operasi perkalian, melainkan bagian dari notasi resmi komponen warna L*, a*, dan b*. CIELAB memisahkan informasi kecerahan dan warna ke dalam tiga komponen, yaitu L*, a*, dan b*.

1. **L*** merepresentasikan tingkat kecerahan piksel, dari gelap hingga terang.
2. **a*** merepresentasikan sumbu warna dari hijau menuju merah.
3. **b*** merepresentasikan sumbu warna dari biru menuju kuning.

Pemisahan komponen kecerahan dan warna membuat CIELAB relatif lebih sesuai untuk menganalisis citra kulit dengan variasi pencahayaan. Pada kondisi pencahayaan yang berubah, komponen L* cenderung menangkap perubahan kecerahan, sedangkan komponen a* dan b* membantu merepresentasikan perbedaan warna. Dalam konteks jerawat, komponen a* bermanfaat untuk menonjolkan area kemerahan pada lesi inflamasi seperti papules dan pustules [15].

Ruang warna CIELAB dipilih dalam penelitian ini karena empat kelas lesi target memiliki karakteristik warna yang berbeda. *Blackheads* cenderung memiliki titik gelap, *Papules* dan *Pustules* memiliki komponen kemerahan, sedangkan *Whiteheads* memiliki area terang yang pucat. Variasi tipe kulit juga perlu

diperhatikan karena warna dasar kulit dapat memengaruhi kontras visual antara lesi dan kulit di sekitarnya [14, 15, 17]. Untuk penyederhanaan penulisan, ruang warna CIELAB selanjutnya disebut sebagai LAB.

2.5.1 Histogram Warna

Histogram warna merupakan representasi distribusi frekuensi piksel terhadap nilai warna pada suatu *color space* [18]. Histogram dihitung dengan membagi rentang nilai setiap kanal ke dalam sejumlah interval yang disebut bin, kemudian menghitung jumlah piksel yang berada pada setiap bin. Representasi histogram digunakan agar citra lesi dapat diubah menjadi vektor fitur numerik berdimensi tetap sebelum diproses oleh *Random Forest* [18, 22].

Dalam penelitian ini, histogram dihitung secara terpisah pada ketiga kanal LAB menggunakan 16 bin per kanal. Penggabungan histogram dari ketiga kanal menghasilkan vektor fitur warna berdimensi $16 \times 3 = 48$. Fitur ini merepresentasikan distribusi warna global lesi tanpa mempertimbangkan posisi piksel [22].

2.6 Ekstraksi Fitur

Ekstraksi fitur merupakan proses mengubah citra digital menjadi representasi numerik yang lebih ringkas dan dapat diproses oleh algoritma *Machine Learning* [18]. *Random Forest* tidak dirancang untuk memproses citra mentah secara langsung seperti CNN, sehingga citra perlu direpresentasikan terlebih dahulu dalam bentuk fitur numerik berdimensi tetap. Pemilihan fitur yang tepat berpengaruh langsung terhadap kemampuan model dalam membedakan kelas yang memiliki kemiripan visual [18, 20].

Dalam penelitian ini, ekstraksi fitur dilakukan pada citra lesi hasil *crop* yang telah diubah ukurannya menjadi 64×64 piksel [18, 22, 15]. Fitur yang digunakan berjumlah 63 dimensi dan terdiri dari empat kelompok fitur sebagaimana ditunjukkan pada Tabel 2.2.

Tabel 2.2. Kelompok fitur dan dimensinya

No	Kelompok Fitur	Dimensi	Target Informasi
1	LAB Color Histogram	48	Distribusi warna global lesi
2	Center-vs-Periphery LAB	9	Perbedaan warna pusat dan pinggir lesi
3	Brightest Point Detection	3	Titik paling terang sebagai indikator area nanah
4	Darkest Point Detection	3	Titik paling gelap sebagai indikator oksidasi
Total		**63**	

1. **LAB Color Histogram (48 dimensi).** Histogram warna dihitung secara terpisah pada kanal L, a, dan b menggunakan 16 bin per kanal. Nilai histogram dinormalisasi terhadap total piksel citra agar tidak bergantung pada ukuran *crop* [18, 22]. Perhitungan histogram ditunjukkan pada Rumus 2.1.

$$h_c[k] = \frac{\sum_{(x,y)} \mathbf{1}[I_c(x,y) \in \text{bin}_k]}{W \times H} \tag{2.1}$$

di mana $h_c[k]$ adalah frekuensi relatif bin ke-k pada kanal $c \in \{L,a,b\}$, $I_c(x,y)$ adalah nilai piksel pada posisi (x,y), dan $W \times H$ adalah jumlah piksel citra.

2. **Center-vs-Periphery LAB (9 dimensi).** Fitur ini mengukur perbedaan karakteristik warna antara zona pusat dan zona pinggir lesi. Citra 64×64 piksel dibagi menjadi dua zona menggunakan mask lingkaran dengan radius 35% dari ukuran citra [18, 15]. Selisih rata-rata LAB antara pusat dan pinggir dihitung sebagaimana ditunjukkan pada Rumus 2.2.

$$\Delta c = \bar{c}_{center} - \bar{c}_{periphery}, \quad c \in \{L,a,b\} \tag{2.2}$$

Selisih rata-rata ketiga kanal menghasilkan 3 dimensi, sedangkan standar deviasi pada zona pusat dan pinggir menghasilkan 6 dimensi. Total fitur ini berjumlah 9 dimensi. Fitur ini dirancang untuk menangkap pola pusat dan pinggir lesi, terutama pada pustules yang memiliki area terang di pusat dan area kemerahan di sekitarnya [3, 15].

3. **Brightest Point Detection (3 dimensi).** Fitur ini mengidentifikasi piksel dengan nilai kecerahan tertinggi pada kanal L. Tiga nilai yang diekstraksi adalah nilai L maksimum, selisih L maksimum terhadap rata-rata L, dan jarak piksel paling terang dari pusat citra [18, 15]. Perhitungan fitur titik paling terang ditunjukkan pada Rumus 2.3.

$$L_{\max} = \max_{(x,y)} L(x,y), \quad c_{bright} = L_{\max} - \bar{L}, \quad d_{bright} = \frac{\sqrt{(x_b - 32)^2 + (y_b - 32)^2}}{32} \tag{2.3}$$

Fitur ini digunakan untuk memperkuat sinyal area terang, terutama pada pustules yang memiliki titik putih atau kekuningan [3, 15].

4. **Darkest Point Detection (3 dimensi).** Fitur ini mengidentifikasi piksel dengan nilai kecerahan terendah pada kanal L. Tiga nilai yang diekstraksi adalah nilai L minimum, selisih rata-rata L terhadap L minimum, dan jarak piksel paling gelap dari pusat citra [18, 15]. Perhitungan fitur titik paling gelap ditunjukkan pada Rumus 2.4.

$$L_{\min} = \min_{(x,y)} L(x,y), \quad c_{dark} = \bar{L} - L_{\min}, \quad d_{dark} = \frac{\sqrt{(x_d - 32)^2 + (y_d - 32)^2}}{32} \tag{2.4}$$

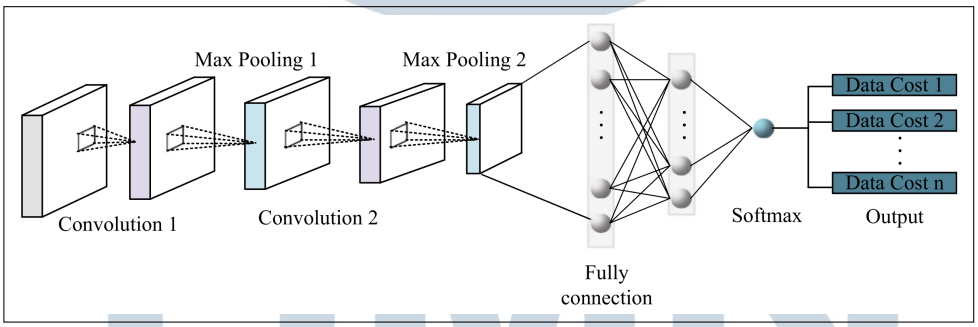
Fitur ini digunakan untuk memperkuat sinyal titik gelap pada blackheads [3, 15].

Penggabungan keempat kelompok fitur tersebut menghasilkan vektor fitur berdimensi 63 yang digunakan sebagai masukan model *Random Forest*.

2.7 Convolutional Neural Network

Convolutional Neural Network (CNN) merupakan salah satu arsitektur *Deep Learning* yang banyak digunakan pada pengolahan citra. CNN dirancang untuk mempelajari pola visual langsung dari matriks piksel citra melalui beberapa tahapan, seperti *convolution*, fungsi aktivasi, *pooling*, dan *fully connected layer*. Zhao dkk. menjelaskan bahwa CNN secara umum terdiri dari lapisan *convolution*, *pooling*, fungsi aktivasi, *fully connected layer*, dan lapisan keluaran untuk menghasilkan prediksi kelas [23]. Pada bidang klasifikasi citra kulit, CNN telah digunakan untuk membantu pengenalan kondisi atau penyakit kulit berbasis citra [5, 2, 6].

Secara umum, CNN terdiri dari dua bagian utama, yaitu tahap ekstraksi fitur dan tahap klasifikasi. Tahap ekstraksi fitur dilakukan melalui *convolution layer*, fungsi aktivasi, dan *pooling layer* untuk menghasilkan *feature map*. Setelah itu, hasil ekstraksi fitur diubah menjadi vektor melalui proses *flattening* dan diteruskan ke *fully connected layer* untuk menghasilkan prediksi kelas. Arsitektur umum CNN ditunjukkan pada Gambar 2.7.



Gambar 2.7. Arsitektur umum *Convolutional Neural Network*
Sumber: Artikel jurnal Zhao dkk. [23]

2.7.1 Convolution Layer

Convolution layer merupakan lapisan utama pada CNN yang berfungsi mengekstraksi pola lokal dari citra. Lapisan ini menggunakan *filter* atau *kernel* berukuran kecil yang digeser pada area citra masukan. Operasi *convolution* dilakukan dengan mengalikan nilai *kernel* dengan nilai piksel pada area citra yang bersesuaian, kemudian seluruh hasil perkalian dijumlahkan [23, 18]. Hasil dari proses tersebut disebut *feature map*, yaitu matriks baru yang merepresentasikan respons fitur tertentu pada citra.

Ilustrasi proses *convolution* dapat dijelaskan menggunakan matriks citra sederhana. Misalkan terdapat citra masukan berukuran 5×5 sebagai berikut:

$$I = \begin{bmatrix} 9 & 4 & 1 & 2 & 2 \\ 1 & 1 & 1 & 0 & 4 \\ 1 & 2 & 1 & 0 & 2 \\ 1 & 0 & 0 & 2 & 2 \\ 9 & 6 & 7 & 4 & 4 \end{bmatrix}$$

Kernel yang digunakan berukuran 3×3 sebagai berikut:

$$K = \begin{bmatrix} 0 & 2 & 1 \\ 4 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Pada posisi pertama, *kernel* diletakkan pada area kiri atas citra masukan. Area citra yang diproses adalah:

$$I_{0:3,0:3} = \begin{bmatrix} 9 & 4 & 1 \\ 1 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix}$$

Setiap nilai pada area citra dikalikan dengan nilai *kernel* pada posisi yang sama. Nilai pertama pada *output array* dihitung sebagai berikut:

$$\begin{aligned} Output[0][0] = &(9 \times 0) + (4 \times 2) + (1 \times 1) \\ &+ (1 \times 4) + (1 \times 1) + (1 \times 0) \\ &+ (1 \times 1) + (2 \times 0) + (1 \times 1) \end{aligned}$$

Hasil setiap perkalian dapat dijabarkan menjadi:

$$Output[0][0] = 0 + 8 + 1 + 4 + 1 + 0 + 1 + 0 + 1$$

Sehingga nilai akhirnya adalah:

$$Output[0][0] = 16$$

Nilai 16 tersebut ditempatkan pada posisi pertama *output array*. Setelah itu, *kernel* digeser satu kolom ke kanan karena *stride* yang digunakan bernilai 1.

Proses perkalian dan penjumlahan yang sama dilakukan pada seluruh posisi hingga menghasilkan *feature map* akhir sebagai berikut:

$$I * K = \begin{bmatrix} 16 & 11 & 13 \\ 10 & 13 & 10 \\ 25 & 12 & 15 \end{bmatrix}$$

Sebagai contoh, nilai 11 pada posisi $Output[0][1]$ diperoleh dari area citra berikut:

$$\begin{bmatrix} 4 & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 1 & 0 \end{bmatrix}$$

Kemudian area tersebut dikalikan dengan *kernel* yang sama, sehingga:

$$\begin{aligned} Output[0][1] =& (4 \times 0) + (1 \times 2) + (2 \times 1) \\ &+ (1 \times 4) + (1 \times 1) + (0 \times 0) \\ &+ (2 \times 1) + (1 \times 0) + (0 \times 1) = 11 \end{aligned}$$

Dengan cara yang sama, seluruh nilai pada *feature map* dihitung dari area 3×3 yang berbeda pada citra masukan. Hal ini menunjukkan bahwa *convolution* bekerja dengan membaca pola lokal secara bertahap, bukan langsung membaca keseluruhan citra sekaligus. Pola lokal tersebut dapat berupa tepi, tekstur, warna, atau bentuk kecil yang kemudian digunakan oleh layer berikutnya.

Secara umum, ukuran *feature map* hasil *convolution* dipengaruhi oleh ukuran citra masukan, ukuran *kernel*, nilai *padding*, dan nilai *stride*. Jika citra masukan berukuran $N \times N$, *kernel* berukuran $F \times F$, *padding* sebesar P, dan *stride* sebesar S, maka ukuran *output* dapat dihitung menggunakan Persamaan 2.5 [23, 18].

$$O = \frac{N - F + 2P}{S} + 1 \tag{2.5}$$

Pada Persamaan 2.5, O adalah ukuran sisi *feature map* keluaran, N adalah ukuran sisi citra masukan, F adalah ukuran sisi *kernel*, P adalah jumlah *padding*, dan S adalah nilai *stride*. Pada contoh matriks sebelumnya, citra masukan berukuran 5×5, *kernel* berukuran 3×3, tidak menggunakan *padding*, dan *stride* bernilai 1. Dengan demikian, ukuran keluaran dihitung sebagai berikut:

$$O = \frac{5-3+2(0)}{1} + 1 = 3$$

Hasil tersebut menunjukkan bahwa ukuran *feature map* adalah 3×3. Ukuran ini sesuai dengan hasil matriks *convolution* yang telah ditunjukkan sebelumnya.

2.7.2 Activation Function dan Pooling Layer

Setelah proses *convolution*, nilai pada *feature map* biasanya diteruskan ke fungsi aktivasi. Fungsi aktivasi digunakan untuk menambahkan sifat nonlinier agar model dapat mempelajari pola yang lebih kompleks. Salah satu fungsi aktivasi yang umum digunakan adalah *Rectified Linear Unit* (ReLU). ReLU bekerja dengan mempertahankan nilai positif dan mengubah nilai negatif menjadi nol [23].

Secara sederhana, fungsi ReLU dapat dituliskan sebagai berikut:

$$ReLU(x) = \max(0, x)$$

Sebagai contoh, jika terdapat nilai $x = -3$, maka hasil ReLU adalah 0. Jika terdapat nilai $x = 5$, maka hasil ReLU tetap 5. Dengan demikian, nilai negatif yang dianggap kurang relevan dapat ditekan, sedangkan respons fitur yang kuat tetap dipertahankan.

Setelah aktivasi, *pooling layer* digunakan untuk mereduksi ukuran *feature map*. Zhao dkk. menjelaskan bahwa *pooling* membagi *feature map* menjadi beberapa area lokal, kemudian menghasilkan satu nilai keluaran dari setiap area tersebut [23]. Tujuan *pooling* adalah membuat representasi fitur menjadi lebih ringkas dan mengurangi jumlah komputasi pada layer berikutnya.

Salah satu metode yang umum digunakan adalah *max pooling*. Pada *max pooling*, nilai terbesar dari setiap blok kecil pada *feature map* dipilih sebagai keluaran. Misalkan terdapat *feature map* berukuran 4×4 sebagai berikut:

$$F = \begin{bmatrix} 2 & 4 & 2 & 1 \\ 4 & 4 & 3 & 2 \\ 2 & 2 & 3 & 5 \\ 1 & 3 & 2 & 4 \end{bmatrix}$$

Jika digunakan *max pooling* berukuran 2×2 dengan *stride* 2, maka *feature map* dibagi menjadi empat blok 2×2. Blok pertama adalah:

$$\begin{bmatrix} 2 & 4 \\ 4 & 4 \end{bmatrix}$$

Nilai maksimum pada blok tersebut adalah 4, sehingga nilai 4 menjadi elemen pertama pada hasil *pooling*. Blok kedua adalah:

$$\begin{bmatrix} 2 & 1 \\ 3 & 2 \end{bmatrix}$$

Nilai maksimum pada blok kedua adalah 3. Blok ketiga adalah:

$$\begin{bmatrix} 2 & 2 \\ 1 & 3 \end{bmatrix}$$

Nilai maksimum pada blok ketiga adalah 3. Blok keempat adalah:

$$\begin{bmatrix} 3 & 5 \\ 2 & 4 \end{bmatrix}$$

Nilai maksimum pada blok keempat adalah 5. Dengan demikian, hasil *max pooling* adalah:

$$\text{MaxPool}(F) = \begin{bmatrix} 4 & 3 \\ 3 & 5 \end{bmatrix}$$

Hasil tersebut menunjukkan bahwa *pooling* mengubah *feature map* dari ukuran 4×4 menjadi 2×2. Meskipun ukurannya lebih kecil, informasi dengan respons paling dominan tetap dipertahankan. Dalam CNN, proses ini membantu mengurangi beban komputasi dan membuat fitur yang dipelajari menjadi lebih ringkas sebelum masuk ke layer berikutnya.

2.7.3 Peran CNN dalam Penelitian Ini

Dalam penelitian ini, CNN tidak digunakan sebagai model klasifikasi utama yang dilatih dari awal. CNN dibahas sebagai landasan teori karena arsitektur deteksi objek modern, termasuk keluarga YOLO, dibangun dari prinsip ekstraksi fitur visual berbasis *convolution*. YOLO26 digunakan untuk mendeteksi lokasi lesi jerawat dan menghasilkan *bounding box*, sedangkan klasifikasi lanjutan dilakukan menggunakan *Random Forest* berbasis fitur LAB 63 dimensi.

2.8 Google Colaboratory

Google Colaboratory atau Google Colab merupakan layanan komputasi awan berbasis Jupyter Notebook yang dapat digunakan untuk menjalankan kode Python melalui *web browser*. Platform ini menyediakan akses terhadap GPU sehingga dapat membantu proses pelatihan model *Machine Learning* dan *Deep Learning* yang membutuhkan sumber daya komputasi lebih besar [24].

Dalam penelitian ini, Google Colab digunakan untuk melatih model YOLO26 karena proses pelatihan model deteksi objek membutuhkan GPU dan memori yang lebih besar dibandingkan perangkat lokal. Hasil pelatihan berupa file bobot model disimpan ke Google Drive agar tetap tersedia setelah sesi Colab berakhir [24].

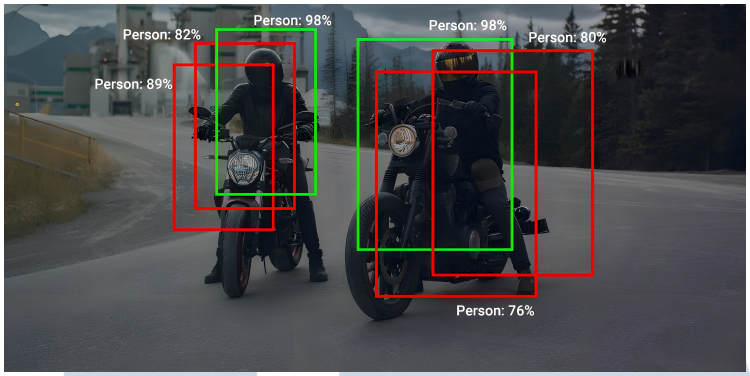
2.9 YOLO26 (You Only Look Once 26)

YOLO atau *You Only Look Once* merupakan keluarga model deteksi objek yang melakukan lokalisasi dan klasifikasi objek dalam satu tahap inferensi. Berbeda dengan pendekatan dua tahap, YOLO memproses citra secara langsung dan menghasilkan *bounding box*, label kelas, serta *confidence* score dalam satu alur. Karakteristik ini membuat YOLO banyak digunakan pada aplikasi yang membutuhkan deteksi cepat [7, 25, 26].

YOLO26 merupakan salah satu varian YOLO yang dirancang dengan pendekatan deteksi end-to-end dan berupaya mengurangi ketergantungan terhadap *post-processing* berbasis *Non Maximum Suppression* (NMS) [7, 25]. Dalam penelitian ini, YOLO26 digunakan sebagai model deteksi awal untuk melokalisasi lesi jerawat pada citra wajah. Hasil deteksi tersebut menjadi dasar pembentukan *crop* lesi yang selanjutnya dianalisis oleh model klasifikasi lanjutan.

2.9.1 Arsitektur Bebas NMS

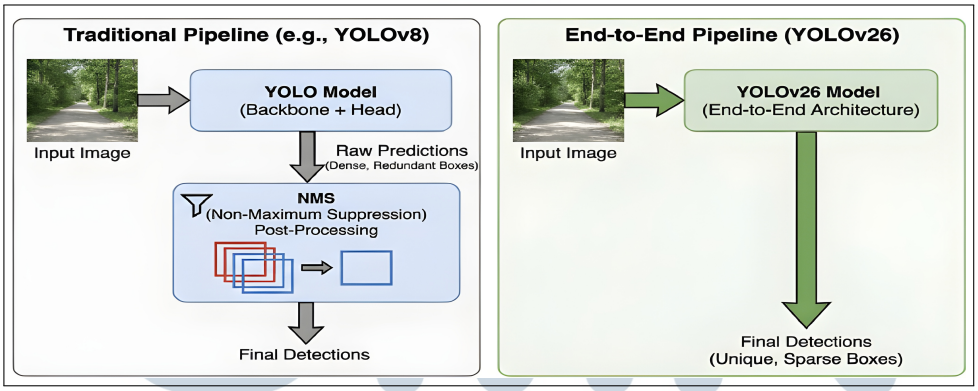
Pada model deteksi objek konvensional, NMS digunakan untuk menyaring *bounding box* yang saling tumpang tindih. NMS memilih *bounding box* dengan *confidence* tertinggi, kemudian menghapus *bounding box* lain yang memiliki nilai *Intersection over Union* (IoU) melebihi ambang tertentu. Meskipun efektif, proses ini dapat menambah latensi dan bergantung pada pemilihan *threshold* [27, 7]. Contoh kelemahan deteksi kotak pembatas yang saling tumpang tindih pada arsitektur YOLO terdahulu ditunjukkan pada Gambar 2.8.



Gambar 2.8. Kelemahan deteksi kotak pembatas yang tumpang tindih pada arsitektur YOLO terdahulu

Sumber: Artikel web Analytics Vidhya [27]

YOLO26 menawarkan pendekatan yang mengurangi kebutuhan terhadap NMS dengan menghasilkan prediksi akhir secara lebih langsung melalui jaringan deteksi. Pendekatan ini berpotensi membuat alur inferensi lebih sederhana dan lebih konsisten pada berbagai kondisi jumlah objek dalam citra [7, 26]. Perbandingan alur kerja deteksi YOLO terdahulu dan YOLO26 yang bebas NMS ditunjukkan pada Gambar 2.9.



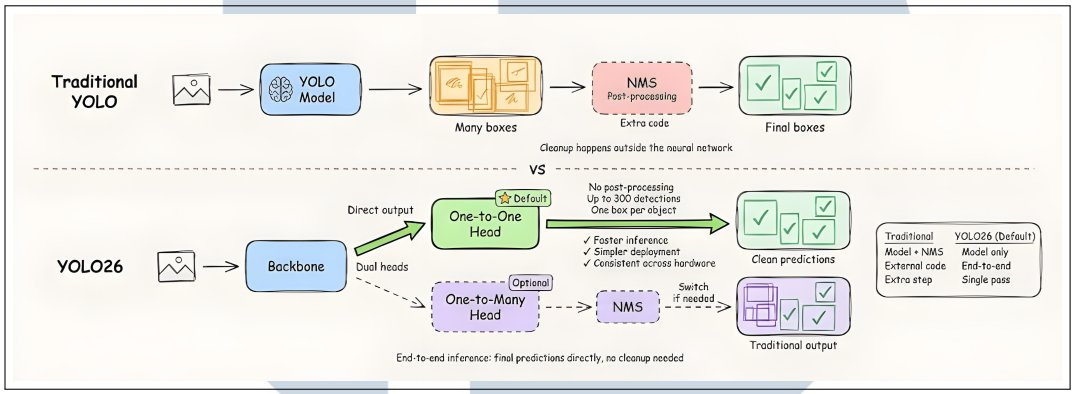
Gambar 2.9. Perbandingan alur kerja deteksi YOLO terdahulu dan YOLO26 yang bebas NMS

Sumber: Artikel ilmiah Chakrabarty [7]

2.9.2 Native One-to-One Prediction

Untuk mengurangi kebutuhan terhadap NMS, YOLO26 menggunakan mekanisme native one-to-one prediction. Pada mekanisme ini, model diarahkan agar setiap objek dipasangkan dengan satu prediksi utama. Dengan demikian, jumlah prediksi ganda yang harus disaring pada tahap akhir dapat dikurangi [7, 25].

Beberapa literatur juga menjelaskan bahwa YOLO26 menggunakan komponen pelatihan tambahan untuk menjaga stabilitas konvergensi, seperti pengaturan head prediksi dan strategi optimasi tertentu [7]. Namun, penelitian ini berfokus pada pemanfaatan YOLO26 sebagai model deteksi objek melalui implementasi Ultralytics, bukan pada modifikasi internal arsitektur YOLO26. Perbandingan alur inferensi YOLO terdahulu dan YOLO26 dengan kepala prediksi satu ke satu ditunjukkan pada Gambar 2.10.



Gambar 2.10. Perbandingan alur inferensi YOLO terdahulu dan YOLO26 dengan kepala prediksi satu ke satu

Sumber: Artikel web Chawla [28]

2.9.3 Peran YOLO26 dalam Penelitian

Dalam penelitian ini, YOLO26 memiliki dua peran utama. Pertama, YOLO26 digunakan sebagai *detector* untuk melokalisasi lesi jerawat pada citra wajah. Kedua, hasil deteksi YOLO26 digunakan untuk menghasilkan *crop* lesi yang menjadi masukan proses ekstraksi fitur dan klasifikasi lanjutan. Nilai *confidence* dari YOLO26 juga digunakan dalam mekanisme *confidence-based routing* untuk menentukan apakah label awal diterima langsung atau diteruskan ke *Random Forest* [7, 29].

2.10 Algoritma Random Forest

Random Forest merupakan algoritma *Machine Learning* berbasis *ensemble* yang membangun banyak *decision tree* dan menggabungkan hasil prediksi setiap pohon untuk menghasilkan keputusan akhir. Pada tugas klasifikasi, setiap pohon memberikan prediksi kelas, kemudian kelas akhir ditentukan melalui *majority voting* [20].

Random Forest dipilih karena mampu bekerja dengan fitur numerik terstruktur, relatif stabil terhadap data nonlinier, dan lebih mudah diinterpretasikan dibandingkan beberapa model *Deep Learning*. Dalam penelitian ini, *Random Forest* digunakan untuk mengklasifikasikan vektor fitur LAB 63 dimensi yang diekstraksi dari citra lesi hasil *crop* [20, 22].

2.10.1 Cara Kerja Decision Tree

Decision Tree merupakan model klasifikasi berbentuk struktur pohon yang terdiri dari simpul akar, simpul internal, cabang, dan simpul daun. Setiap simpul internal berisi aturan keputusan terhadap salah satu fitur, misalnya apakah nilai suatu fitur lebih kecil atau sama dengan nilai ambang tertentu. Data yang memenuhi aturan akan diarahkan ke cabang kiri, sedangkan data yang tidak memenuhi aturan diarahkan ke cabang kanan. Proses ini berulang hingga data mencapai simpul daun, yaitu bagian akhir pohon yang menyimpan kelas prediksi [20].

Pada tahap pelatihan, *Decision Tree* mencari fitur dan ambang batas yang mampu memisahkan data menjadi kelompok yang lebih homogen. Kualitas pemisahan tersebut dapat dihitung menggunakan kriteria seperti *Gini Impurity*. Pemisahan yang baik adalah pemisahan yang menghasilkan simpul anak dengan nilai impurity lebih rendah dibandingkan simpul induk. Dengan demikian, proses pembentukan pohon dilakukan secara bertahap dari simpul akar hingga kondisi penghentian tertentu terpenuhi, misalnya batas kedalaman maksimum atau jumlah minimum sampel pada simpul [20].

Salah satu hyperparameter penting pada *Decision Tree* adalah **max_depth**. Nilai **max_depth** mengatur kedalaman maksimum pohon. Pohon yang terlalu dalam dapat mempelajari detail data latih secara berlebihan sehingga rentan terhadap *overfitting*. Sebaliknya, pohon yang terlalu dangkal dapat kurang mampu menangkap pola data sehingga rentan terhadap *underfitting*. Oleh karena itu, kedalaman pohon perlu diatur agar model memiliki keseimbangan antara kemampuan belajar dan kemampuan generalisasi [20].

2.10.2 Cara Kerja Random Forest

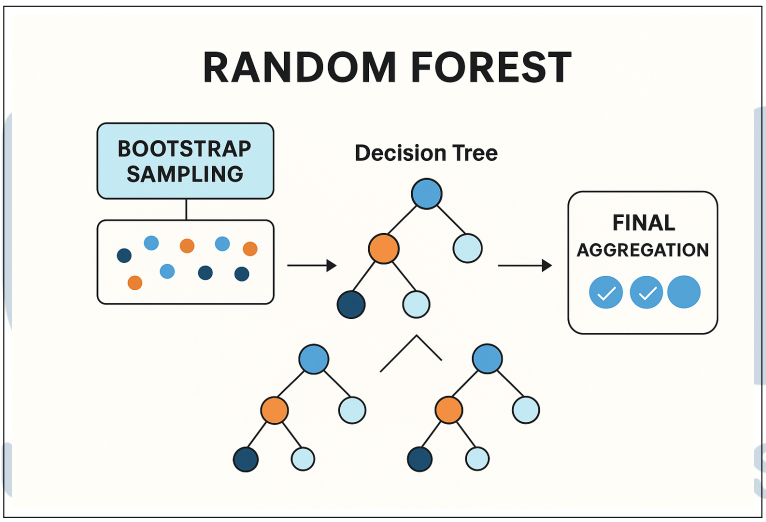
Random Forest mengembangkan konsep *Decision Tree* dengan membangun banyak pohon keputusan. Jika *Decision Tree* tunggal hanya menghasilkan prediksi dari satu struktur pohon, *Random Forest* menghasilkan prediksi dari banyak pohon

yang dilatih menggunakan variasi data dan variasi fitur. Setiap pohon memberikan satu suara kelas, kemudian kelas dengan jumlah suara terbanyak menjadi prediksi akhir melalui mekanisme *majority voting* [20].

Perbedaan utama *Random Forest* dibandingkan *Decision Tree* tunggal terletak pada penggunaan teknik *ensemble*. Pada *Decision Tree* tunggal, kesalahan pada struktur pohon dapat langsung memengaruhi prediksi akhir. Pada *Random Forest*, prediksi akhir tidak bergantung pada satu pohon saja, tetapi merupakan hasil agregasi dari banyak pohon. Hal ini membuat model lebih stabil dan dapat mengurangi risiko *overfitting* dibandingkan penggunaan satu pohon keputusan [20].

2.10.3 Pengambilan Sampel Bootstrap

Pada tahap pelatihan, *Random Forest* menggunakan teknik *bootstrap sampling*. Teknik ini mengambil sampel data secara acak dengan pengembalian untuk membentuk subset pelatihan yang berbeda pada setiap pohon. Perbedaan subset data membuat setiap pohon mempelajari pola yang berbeda, sehingga model *ensemble* menjadi lebih stabil [20]. Mekanisme pengambilan sampel bootstrap pada pelatihan data ditunjukkan pada Gambar 2.11.



Gambar 2.11. Ilustrasi mekanisme pengambilan sampel bootstrap pada pelatihan data
Sumber: Artikel jurnal Salman dkk. [20]

2.10.4 Pemilihan Fitur Acak pada Percabangan Pohon

Selain mengambil subset data, *Random Forest* juga menerapkan pemilihan fitur acak pada proses *node splitting*. Setiap pohon tidak selalu mengevaluasi

seluruh fitur, melainkan hanya sebagian fitur yang dipilih secara acak. Strategi ini mengurangi korelasi antar pohon dan membantu menekan risiko *overfitting* [20].

Setelah kandidat fitur dipilih, kualitas pemisahan pada pohon keputusan dapat diukur menggunakan *Gini Impurity*. Nilai Gini yang lebih kecil menunjukkan bahwa data pada suatu simpul cenderung lebih homogen terhadap kelas tertentu. Perhitungan *Gini Impurity* ditunjukkan pada Rumus 2.6.

$$Gini(t) = 1 - \sum_{i=1}^{K} p_i^2 \tag{2.6}$$

di mana K adalah jumlah kelas dan p_i adalah proporsi sampel kelas ke-i pada simpul t [20].

Pada proses pemisahan simpul, model menghitung nilai impurity pada simpul induk dan membandingkannya dengan impurity gabungan pada simpul anak kiri dan kanan. Nilai impurity gabungan dihitung secara berbobot berdasarkan jumlah sampel pada masing-masing simpul anak. Perhitungan *Gini Split* ditunjukkan pada Rumus 2.7.

$$Gini_{split} = \frac{n_{left}}{n} Gini_{left} + \frac{n_{right}}{n} Gini_{right} \tag{2.7}$$

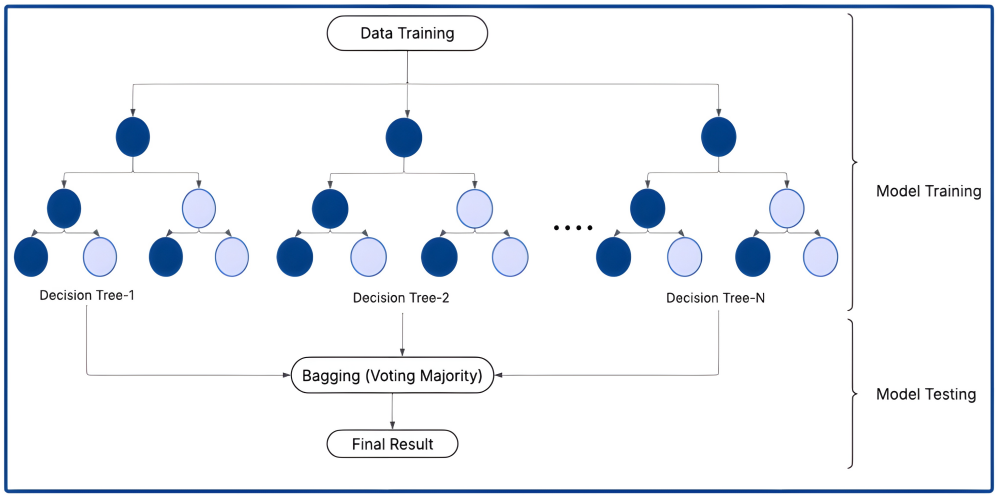
di mana n adalah jumlah sampel pada simpul induk, n_{left} adalah jumlah sampel pada simpul anak kiri, dan n_{right} adalah jumlah sampel pada simpul anak kanan. Pemisahan yang dipilih adalah pemisahan yang menghasilkan penurunan impurity terbesar, yaitu selisih antara nilai Gini pada simpul induk dan nilai *Gini Split*. Semakin besar penurunan impurity, semakin baik pemisahan tersebut dalam membuat kelompok data menjadi lebih homogen [20].

2.10.5 Agregasi dan Pemungutan Suara

Setelah seluruh pohon menghasilkan prediksi, *Random Forest* melakukan agregasi keputusan menggunakan *majority voting*. Kelas yang memperoleh suara terbanyak dari seluruh pohon menjadi prediksi akhir model. Perhitungan prediksi akhir berdasarkan *majority voting* ditunjukkan pada Rumus 2.8.

$$\hat{y} = \text{mode}(h_1(x), h_2(x), \ldots, h_T(x)) \tag{2.8}$$

di mana $\hat{y}$ adalah kelas prediksi akhir, $h_i(x)$ adalah hasil prediksi pohon ke-i, dan T adalah jumlah pohon dalam *Random Forest* [20]. Mekanisme agregasi pemungutan suara pada *Random Forest* ditunjukkan pada Gambar 2.12.



Gambar 2.12. Mekanisme agregasi pemungutan suara pada arsitektur *Random Forest*
Sumber: Artikel jurnal Maitimu dkk. [11]

2.10.6 Hyperparameter dan Hyperparameter Sweep

Hyperparameter merupakan konfigurasi model yang ditentukan sebelum pelatihan dimulai. Pada *Random Forest*, beberapa hyperparameter penting yang digunakan dalam penelitian ini antara lain jumlah pohon, kedalaman maksimum pohon, jumlah minimum sampel pada daun, jumlah minimum sampel untuk pemisahan simpul, dan pembobotan kelas. Pemilihan hyperparameter yang tepat dapat membantu menyeimbangkan kemampuan model dalam mempelajari pola dan menghindari *overfitting* [20].

1. **n_estimators** adalah jumlah pohon dalam *forest*.
2. **max_depth** adalah kedalaman maksimum setiap pohon.
3. **min_samples_leaf** adalah jumlah minimum sampel pada daun pohon.
4. **min_samples_split** adalah jumlah minimum sampel yang dibutuhkan agar suatu simpul dapat dipecah.
5. **criterion** adalah fungsi pengukuran kualitas pemisahan, seperti *Gini Impurity*.

6. **class_weight** adalah pengaturan bobot kelas untuk membantu menangani distribusi data yang tidak seimbang.

Dalam penelitian ini, pemilihan konfigurasi *Random Forest* dilakukan melalui *hyperparameter sweep*. Setiap kombinasi parameter diuji pada data validasi, kemudian konfigurasi terbaik dipilih berdasarkan performa validasi dan indikasi generalisasi model [20].

Karena distribusi data antar kelas tidak seimbang, konfigurasi *Random Forest* juga dapat menggunakan pembobotan kelas *class_weight="balanced"*. Pembobotan ini memberi bobot lebih besar pada kelas dengan jumlah sampel lebih sedikit [20, 29]. Perhitungan bobot kelas seimbang ditunjukkan pada Rumus 2.9.

$$w_j = \frac{N}{K \times n_j} \tag{2.9}$$

di mana w_j adalah bobot kelas ke-j, N adalah jumlah seluruh data pelatihan, K adalah jumlah kelas, dan n_j adalah jumlah data pada kelas ke-j [20, 29].

Pada data dengan distribusi kelas yang tidak seimbang, penyeimbangan data perlu diperlakukan sebagai strategi pelatihan yang harus diuji secara empiris. Penyeimbangan kelas tidak selalu menghasilkan peningkatan performa pada setiap kondisi. Ljubobratovic dkk. membandingkan *Random Forest* pada data asli, data hasil SMOTE, dan data hasil ROSE. Pada salah satu skenario, data asli dan data hasil SMOTE menghasilkan akurasi yang sama, sedangkan data hasil ROSE menghasilkan akurasi yang lebih rendah [30]. Oleh karena itu, penyeimbangan data tidak dapat diasumsikan selalu memperbaiki performa model, terutama apabila distribusi data validasi dan pengujian perlu dipertahankan sesuai kondisi asli.

Jumlah kombinasi yang diuji pada proses *hyperparameter sweep* bergantung pada banyaknya variasi nilai dari setiap hyperparameter. Perhitungan jumlah kombinasi *sweep* ditunjukkan pada Rumus 2.10.

$$N_{sweep} = |E| \times |D| \times |L| \tag{2.10}$$

di mana E adalah himpunan nilai **n_estimators**, D adalah himpunan nilai **max_depth**, dan L adalah himpunan nilai **min_samples_leaf**. Selain performa validasi, selisih performa antara data latih dan data uji dapat digunakan sebagai indikasi *overfitting* [20, 29]. Perhitungan *gap* akurasi ditunjukkan pada Rumus 2.11.

$$Gap = Accuracy_{train} - Accuracy_{test} \tag{2.11}$$

2.10.7 Keunggulan Random Forest pada Pemodelan Hybrid Berbasis Fitur

Random Forest relevan digunakan pada pemodelan *hybrid* karena dapat memproses fitur terstruktur yang dihasilkan dari tahap ekstraksi fitur. Pada penelitian ini, YOLO26 bertugas melokalisasi area lesi, sedangkan *Random Forest* bertugas mengklasifikasikan *crop* lesi berdasarkan fitur LAB. Pembagian tugas tersebut memungkinkan proses klasifikasi dilakukan dengan fitur yang lebih ringkas dan spesifik terhadap karakteristik warna lesi [31, 9].

2.11 Confidence-Based Routing

Confidence-Based Routing merupakan mekanisme pengambilan keputusan yang menggunakan nilai *confidence* dari YOLO26 untuk menentukan jalur klasifikasi akhir. Mekanisme ini terinspirasi dari penelitian Luu dkk. yang menggunakan *Random Forest* untuk membantu klasifikasi pada kondisi prediksi YOLO yang kurang yakin [8].

Secara umum, jika *confidence* YOLO26 berada di atas ambang batas tinggi, prediksi YOLO26 dapat diterima langsung sebagai hasil akhir. Jika *confidence* berada pada rentang tertentu, area lesi yang terdeteksi dipotong, diekstraksi fiturnya, kemudian diklasifikasikan ulang oleh *Random Forest*. Jika *confidence* terlalu rendah, deteksi dapat diabaikan untuk mengurangi risiko prediksi palsu [8, 31].

Dalam penelitian ini, mekanisme Confidence-Based Routing digunakan untuk menguji apakah klasifikasi ulang oleh *Random Forest* pada deteksi dengan *confidence* tertentu dapat membantu memperbaiki prediksi YOLO26. Detail ambang batas *confidence* dan alur routing dijelaskan pada Bab Metodologi [8, 9].

2.12 Metrik Evaluasi Model

Evaluasi performa model pada penelitian ini difokuskan pada metrik klasifikasi multi-kelas, yaitu *Precision*, *Recall*, *F1-Score*, *Macro Precision*, *Macro Recall*, dan *Macro F1-Score*. Metrik tersebut digunakan untuk mengevaluasi performa *Random Forest* sebagai model klasifikasi lanjutan pada dua skenario

pembagian *dataset*. Nilai *Accuracy* dan *Macro Accuracy* digunakan sebagai informasi pendukung, bukan sebagai dasar utama penentuan model terbaik. Fokus evaluasi tersebut dipilih karena tujuan penelitian adalah menilai performa klasifikasi jenis lesi jerawat, bukan mengevaluasi metrik deteksi objek secara terpisah.

2.12.1 Confusion Matrix

Confusion Matrix merupakan matriks yang membandingkan label aktual dan label prediksi model. Pada klasifikasi multi-kelas, nilai True Positive (TP), False Positive (FP), False Negative (FN), dan True Negative (TN) dapat dihitung untuk setiap kelas menggunakan pendekatan *one-vs-rest* [29]. Dengan pendekatan ini, satu kelas dianggap sebagai kelas positif, sedangkan kelas lain dianggap sebagai kelas negatif.

2.12.2 Precision

Precision mengukur proporsi prediksi positif yang benar terhadap seluruh data yang diprediksi sebagai kelas tersebut. Pada konteks klasifikasi jenis jerawat, *precision* menunjukkan seberapa tepat model ketika memberikan prediksi suatu kelas [29]. Perhitungan precision ditunjukkan pada Rumus 2.12.

$$Precision = \frac{TP}{TP + FP} \tag{2.12}$$

2.12.3 Recall

Recall mengukur proporsi data aktual suatu kelas yang berhasil ditemukan oleh model. Pada konteks klasifikasi jerawat, *recall* menunjukkan kemampuan model dalam mengenali seluruh sampel dari kelas tertentu [29]. Perhitungan recall ditunjukkan pada Rumus 2.13.

$$Recall = \frac{TP}{TP + FN} \tag{2.13}$$

2.12.4 F1-Score

F1-Score merupakan rata-rata harmonik antara *precision* dan *recall*. Metrik ini digunakan ketika *precision* dan *recall* perlu dipertimbangkan secara bersamaan, terutama pada *dataset* yang tidak seimbang [29]. Perhitungan F1-Score ditunjukkan pada Rumus 2.14.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.14}$$

2.12.5 Accuracy

Accuracy mengukur proporsi seluruh prediksi yang benar terhadap total data yang dievaluasi. Pada pendekatan *one-vs-rest*, *accuracy* dapat dihitung untuk setiap kelas dengan mempertimbangkan TP, TN, FP, dan FN [29]. Perhitungan accuracy ditunjukkan pada Rumus 2.15.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.15}$$

2.12.6 Macro Average

Macro average merupakan rata-rata metrik dari seluruh kelas dengan bobot yang sama. Pada *dataset* yang tidak seimbang, *macro average* penting karena setiap kelas diperlakukan setara tanpa dipengaruhi jumlah sampel [29]. Perhitungan Macro Precision, Macro Recall, Macro F1-Score, dan Macro Accuracy masing-masing ditunjukkan pada Rumus 2.16, Rumus 2.17, Rumus 2.18, dan Rumus 2.19.

$$Macro\ Precision = \frac{1}{N} \sum_{i=1}^{N} Precision_i \tag{2.16}$$

$$Macro\ Recall = \frac{1}{N} \sum_{i=1}^{N} Recall_i \tag{2.17}$$

$$Macro\ F1 = \frac{1}{N}\sum_{i=1}^{N} F1_i \tag{2.18}$$

$$Macro\ Accuracy = \frac{1}{N}\sum_{i=1}^{N} Accuracy_i \tag{2.19}$$

di mana N adalah jumlah kelas, sedangkan $Precision_i$, $Recall_i$, dan $F1_i$ merupakan nilai *precision*, *recall*, dan *F1-Score* untuk kelas ke-*i*, sedangkan $Accuracy_i$ merupakan nilai *accuracy* untuk kelas ke-*i* [29]. Dalam penelitian ini, *Macro F1-Score* digunakan sebagai metrik utama karena mampu menggambarkan keseimbangan *precision* dan *recall* antar kelas tanpa memberi bobot lebih besar pada kelas mayoritas [29].

2.12.7 Support

Support menunjukkan jumlah sampel berdasarkan label aktual pada setiap kelas. Dengan kata lain, *support* tidak menunjukkan jumlah prediksi benar, melainkan jumlah data uji yang memang berasal dari kelas tersebut [29]. Pada pendekatan *one-vs-rest*, *support* untuk kelas ke-*i* dapat dihitung menggunakan Rumus 2.20.

$$Support_i = TP_i + FN_i \tag{2.20}$$

Dalam penelitian ini, *support* digunakan untuk menjelaskan jumlah data uji pada setiap kelas sehingga pembacaan nilai *precision*, *recall*, dan *F1-Score* dapat dikaitkan dengan banyaknya sampel aktual yang dievaluasi pada masing-masing kelas.