BAB II

LANDASAN TEORI

2.1 Landasan Teori

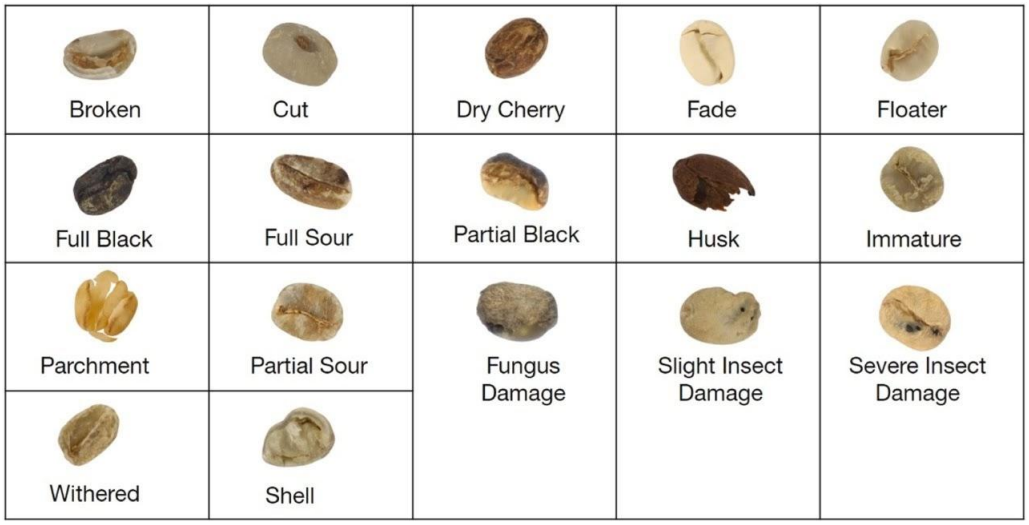
Landasan teori merupakan kumpulan konsep, prinsip, dan hasil penelitian terdahulu yang digunakan sebagai dasar dalam pelaksanaan penelitian (Hanifah dkk., 2025). Dalam penelitian ini, landasan teori mencakup pembahasan mengenai biji kopi dan karakteristik kecacatannya, segmentasi citra, *Segment Anything Model* (SAM), algoritma YOLOv11-Seg, *quantization*, serta metrik evaluasi yang digunakan untuk mengukur performa model.

2.1.1 Biji Kopi (*Coffee Bean*)

Biji kopi merupakan salah satu komoditas pertanian yang populer dan menjadi bahan dasar salah satu minuman yang paling banyak dikonsumsi di dunia (Arfan Astaraja dkk., 2025; Gope dkk., 2024). Secara spesifik, biji kopi hijau (*green coffee bean*) adalah biji bagian dalam yang diekstraksi dari buah kopi setelah melalui proses panen, pengupasan, fermentasi dan pengeringan (Gope dkk., 2024; Oncu, 2025). Kualitas biji kopi dipengaruhi oleh faktor genetik, lingkungan, tempat tanam, serta metode pemrosesan pasca panen yang dilakukan yang dapat berpengaruh terhadap rasa, aroma, dan harga jual produk di pasar global ataupun dalam negeri (Ekowicaksono dkk., 2025; Gope dkk., 2024; Oncu, 2025).

Biji kopi yang berkualitas tinggi sangat penting untuk menghasilkan kopi dengan citra rasa yang baik, sedangkan biji kopi yang cacat dapat menyebabkan bau dan rasa yang tidak sedap serta menurunkan nilai harga jual. Cacat biji kopi, seperti biji hitam atau masam, sering kali mengandung senyawa yang berbahaya dan dapat

menimbulkan risiko kesehatan bagi manusia (Chang & Liu, 2024; Gope dkk., 2024). Sebagai contoh, biji kopi yang terkontaminasi mikroba atau jamur dapat memproduksi *ochratoxin A,* sebuah mikotoksin yang bersifat karsinogenik dan berbahaya bagi ginjal (Arwatchananukul dkk., 2024; Gope dkk., 2024). Contoh lain jenis kecacatan biji kopi terdapat pada gambar 2.1.



Gambar 2.1. Jenis Kecacatan Biji Kopi menurut SCA (Arwatchananukul dkk., 2024)

Standar klasifikasi biji kopi secara internasional sering mengacu pada pedoman *Speciality Coffee Association* (SCA) atau SCAA, yang membagi cacat biji kopi menjadi kategori 1 (cacat utama) dan kategori 2 (cacat sekunder) (Arwatchananukul dkk., 2024). Cacat kategori 1 meliputi biji hitam penuh (*full black*), masam penuh (*full sour*), ceri kering (*dried cherry*), kerusakan akibat jamur (*fungus damage*), dan kerusakan akibat serangga, yang tidak boleh ada sama sekali dalam biji kopi yang akan dikonsumsi. Sementara itu, cacat kategori 2 mencakup biji hitam sebagian (*partial black*), masam sebagian (*partial sour*), biji muda

(*immature*), pecah (broken), dan kerusakan ringan akibat serangga (*SCAA Coffee Beans Classification*, 2006; Thai dkk., 2024).

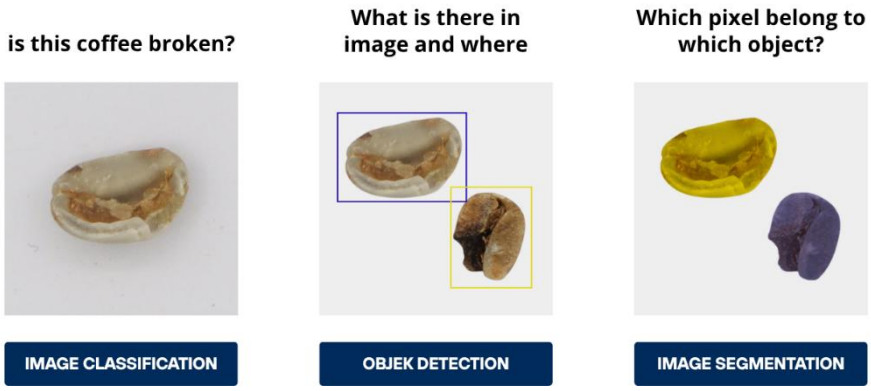
Selain standar SCA, beberapa penelitian lokal (Indonesia) juga mengelompokkan biji kopi berdasarkan bentuk fisiknya . Contohnya, terdapat pada Gambar 2.2, yaitu biji Peaberry (biji tunggal) dan Longberry (biji panjang) sering dipisahkan dari kategori *Premium* dan *Defect* (cacat) (Febriana dkk., 2022). Dalam konteks ini, biji yang dikategorikan sebagai *Defect* adalah biji yang tidak utuh, berlubang, atau berwarna kehitaman yang menyimpang dari standar kualitas visual yang diharapkan.

Gambar 2.2. Gambar Jenis Kecacatan Biji Kopi USK-Coffee: (a) *longberry*, (b) peaberry, (c) premium, dan (d) defect (Febriana dkk., 2022)

Maka dari itu, deteksi dan klasifikasi kualitas biji kopi merupakan langkah yang dapat digunakan dalam rantai produksi untuk menjamin keamanan pangan dan konsistensi rasa bagi konsumen. Metode penyortiran manual yang tradisional dinilai memakan waktu, tenaga, subjektif, dan rentan terhadap kesalahan manusia. Oleh karena itu, penerapan teknologi otomatisasi berbasis kecerdasan buatan (*Deep Learning*) dan visi komputer menjadi solusi yang menjanjikan untuk meningkatkan efisiensi, akurasi, dan objektivitas dalam penilaian kualitas biji kopi (Gope dkk., 2024; Oncu, 2025).

2.1.2 Deteksi Objek (*Object Detection*)

Deteksi objek merupakan salah satu cabang utama dari *computer vision* yang berfungsi untuk mengenali dan menentukan posisi objek tertentu yang telah diberikan label dalam sebuah citra atau video (Manakitsa dkk., 2024). Tugas deteksi objek melibatkan dua proses utama, yaitu lokalisasi objek menggunakan kotak pembatas (*Bounding Box*) dan klasifikasi objek berdasarkan kategori semantik tertentu (Kowalczyk dkk., 2022; Murel & Kavlakoglu, 2025). Dengan menggabungkan kedua proses tersebut, deteksi objek memungkinkan komputer meniru cara manusia dalam memahami lingkungan visual. Oleh karena itu, deteksi objek menjadi fondasi penting dalam pengembangan sistem kecerdasan buatan yang membutuhkan pemahaman visual secara kontekstual. Perbedaan dari klasifikasi, deteksi objek, dan segmentasi gambar terdapat pada gambar 2.3, yang terinspirasi dari (Augmented, 2022).

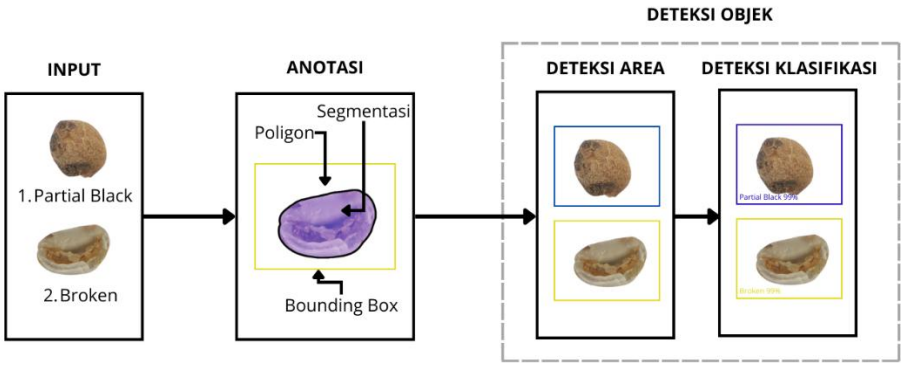


Gambar 2.3. Visualisasi Klasifikasi Citra (*Image Classification*), Deteksi Objek (*Object Detection*), dan Segmentasi Citra (*Image Segmentation*)

Dalam konteks *computer vision,* deteksi objek memiliki perbedaan tugas dengan klasifikasi citra dan segmentasi citra. Klasifikasi citra hanya memberikan satu label pada masing-masing gambar tanpa memperhatikan posisi objek di

dalamnya. Sedangkan, deteksi objek mampu mengidentifikasi banyak objek dalam satu citra beserta lokasi masing-masing. Sementara itu, segmentasi citra bekerja pada tingkat piksel untuk memisahkan objek secara lebih presisi (Manakitsa dkk., 2024). Sehingga deteksi objek sering diposisikan sebagai pendekatan menengah antara klasifikasi dan segmentasi.

Alur kerja deteksi objek dimulai dari persiapan data, di mana data membutuhkan anotasi objek sebagai label objek yang berperan sebagai penentu informasi yang akan dipelajari model . Anotasi yang dibuat dapat berupa *Bounding Box* atau segmentasi. Anotasi atau label yang terdapat pada objek akan digunakan pada tahap pelatihan model yang memindai citra masukan untuk mencari pola fitur yang sesuai dengan data anotasi. Hasil dari pelatihan ini akan menghasilkan klasifikasi objek, di mana setiap objek yang telah berhasil dilokalisasi akan diberikan label identitas tertentu (Murel & Kavlakoglu, 2025). Maka dari itu, deteksi objek tidak hanya mampu menunjukkan “di mana” posisi suatu objek berada, melainkan mampu menjawab mengenai “apa” objek tersebut berdasarkan ciri visual yang telah dipelajari sebelumnya. Cara kerja deteksi objek ditunjukkan pada Gambar 2.4.

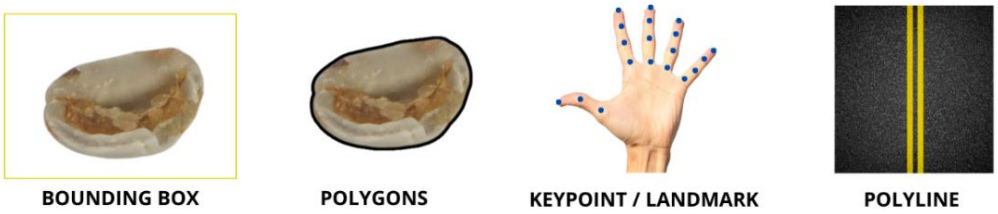


Gambar 2.4. Cara Kerja Deteksi Objek

Dalam pengembangan sistem deteksi objek, pemilihan arsitektur model menjadi aspek penting karena berpengaruh terhadap kemampuan model dalam mengenali dan melokalisasi objek secara akurat. Detektor satu tahap seperti YOLO banyak digunakan karena mampu melakukan deteksi beberapa objek dalam satu citra secara langsung melalui satu alur pemrosesan (Alhashmi & Al-azawi, 2025). Pendekatan ini menjadikan deteksi objek lebih aplikatif untuk berbagai skenario yang membutuhkan respons cepat, termasuk pada citra dengan objek berukuran kecil atau kompleks. Selain itu, integrasi antara proses ekstraksi fitur, prediksi *Bounding Box*, dan klasifikasi kelas dalam satu model memungkinkan deteksi objek dilakukan secara *end-to-end*.

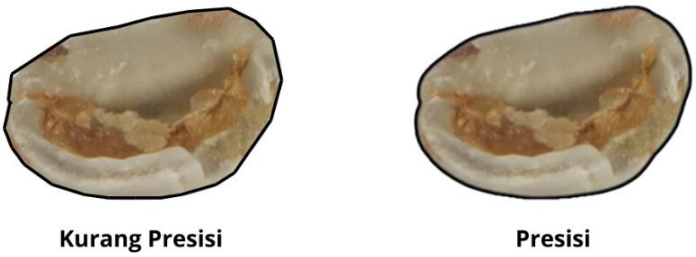
2.1.3 Anotasi (*Annotation*)

Anotasi atau pelabelan citra merupakan proses fundamental dalam pengembangan model kecerdasan buatan yang bertujuan untuk memberikan label atau metadata pada data visual guna memberikan konteks semantik. Proses ini memungkinkan algoritma pembelajaran mesin untuk mengenali pola, mengidentifikasi fitur, dan melakukan prediksi berdasarkan informasi yang telah terstruktur secara manual (Aljabri dkk., 2022). Dalam implementasi praktisnya, anotasi berfungsi sebagai jembatan antara data mentah dengan pemahaman mesin terhadap elemen-elemen di dalam gambar, seperti pada sistem kendaraan otonom, analisis citra medis, hingga pengenalan wajah (Marsyella, 2025).



Gambar 2.5. Teknik Anotasi

Secara teknis, pemilihan metode pelabelan sangat bergantung pada kebutuhan model. Sebagai ilustrasi, Gambar 2.5 menunjukkan empat teknik anotasi yang umum digunakan, yaitu *Bounding Box*, *poligon*, *keypoint* atau *landmark*, dan *polyline* (Barla, 2021). Teknik *Bounding Box* merupakan metode yang paling populer karena efisiensinya dalam melokalisasi objek menggunakan kotak pembatas persegi panjang. Namun, untuk objek dengan kontur yang tidak beraturan, metode poligon menawarkan presisi yang lebih tinggi karena mampu mengikuti tepi objek secara spesifik. Sementara itu, keypoint annotation digunakan untuk mengidentifikasi titik-titik krusial seperti sendi tubuh atau fitur wajah, sedangkan polyline lebih tepat digunakan untuk menandai struktur linear seperti marka jalan atau aliran sungai (Barla, 2021; Marsyella, 2025). Maka dari itu, ketepatan pemilihan jenis anotasi ini akan menentukan performa dan akurasi model dalam memproses data visual sesuai dengan tujuan penelitian yang dikembangkan.



Gambar 2.6. Perbandingan Anotasi Poligon Kurang Presisi dan Presisi

Gambar 2.6 menampilkan perbandingan ilustrasi hasil anotasi poligon dengan tingkat ketelitian yang berbeda. Berdasarkan KBBI, presisi secara etimologis berarti ketepatan atau ketelitian yang merujuk pada tingkat akurasi dalam pengukuran serta penelitian (KBBI, 2025b). Dalam konteks pelabelan objek, hal ini diimplementasikan untuk memastikan bahwa area yang dianotasi benar-benar merepresentasikan objek yang dimaksud secara akurat. Pada anotasi yang kurang presisi, batas pelabelan tidak mengikuti kontur secara akurat sehingga mencakup area di luar objek utama. Sebaliknya, anotasi presisi menunjukkan pelabelan yang rapat dan konsisten dengan tepi objek, sehingga model dapat mempelajari karakteristik bentuk terutama pada objek tidak beraturan.

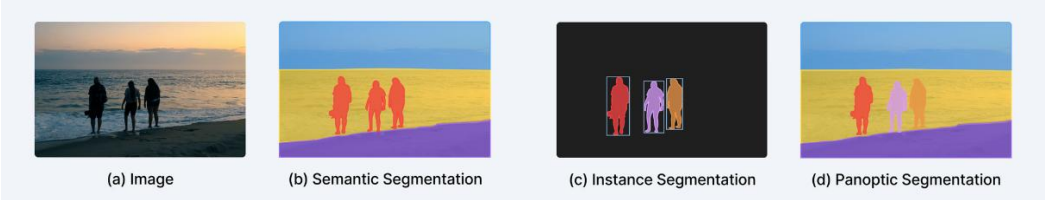
2.1.4 Segmentasi Citra (*Image Segmentation*)

Segmentasi citra merupakan salah satu bagian terpenting dalam *computer vision* selain deteksi objek. Secara teknis, segmentasi didefinisikan sebagai proses klasifikasi tingkat piksel (*pixel-level classification*), di mana setiap piksel dalam citra diberi label kelas tertentu sehingga piksel dengan label yang sama memiliki karakteristik visual yang serupa(Minaee dkk., 2021; Yu dkk., 2023).

Tujuan utama dari proses segmentasi adalah untuk memisahkan *Region of Interest* (ROI) atau objek target dari latar belakang (*background*). Dalam konteks visi komputer, segmentasi menjembatani kesenjangan antara pemrosesan tingkat rendah (seperti deteksi tepi) dan pemrosesan tingkat tinggi (seperti pengenalan objek dan pemahaman adegan). Tidak seperti klasifikasi citra yang memberikan satu label untuk seluruh gambar, segmentasi melakukan klasifikasi pada tingkat

piksel (*pixel-level classification*), di mana setiap piksel dalam citra diberikan label kelas tertentu (Yu dkk., 2023).

Berdasarkan tingkat detail dan tujuan pemisahan objek, segmentasi citra dikategorikan menjadi tiga jenis utama, yakni, *Semantic Segmentation*, *Instance Segmentation*, dan *Panoptic Segmentation* (Borse dkk., 2022). Ilustrasi mengenai perbedaan visual dan karakteristik dari ketiga kategori tersebut disajikan pada Gambar 2.7.



Gambar 2.7. Jenis Segmentasi (Barla, 2022)

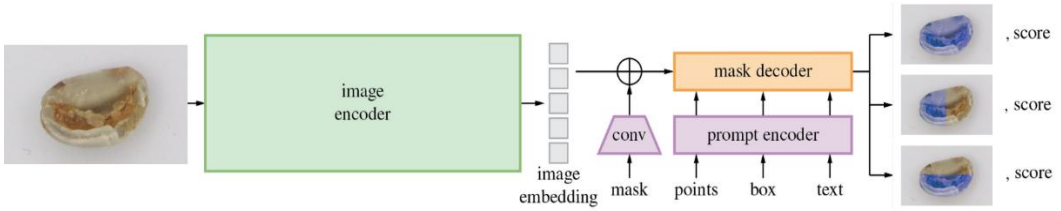
Semantic Segmentation memperlakukan piksel sebagai unit terkecil yang harus diklasifikasikan ke dalam kelas tertentu, contohnya jalan, mobil, dan pejalan kaki. Ciri utama kategori ini adalah tidak adanya pembedaan antara objek-objek individu dari kelas yang sama. *Instance Segmentation* adalah bagian dari penggabungan antara deteksi objek dan segmentasi semantik. Metode ini tidak hanya mengklasifikasikan piksel berdasarkan kategori, tetapi juga membedakan setiap instansi objek secara individu. *Panoptic Segmentation* merupakan gabungan dari *semantic* dan *instance segmentation* untuk memberikan pemahaman adegan yang holistik. Metode *Panoptic Segmentation* membagi objek menjadi dua kategori, yakni *Things* (objek yang dapat dihitung seperti orang dan kendaraan) dan *Stuffs* (latar belakang objek seperti, langit dan jalan) (Barla, 2022; Borse dkk., 2022).

Perkembangan algoritma segmentasi citra secara umum dapat dibagi menjadi dua era utama, yaitu metode klasik dan metode berbasis *deep learning.* Pembagian tersebut didasarkan pada pendekatan yang digunakan dalam memisahkan objek dan latar belakang pada citra digital. Metode klasik menggunakan analisis secara langsung terhadap karakteristik piksel, sedangkan metode *deep learning* memanfaatkan pembelajaran fitur secara otomatis melalui jaringan saraf tiruan. Perkembangan ini menunjukan pergeseran pradigma dari pendekatan berbasis aturan (*rule-based*) menuju pendekatan berbasis (*data-driven*) seiring meningkatnya kompleksitas objek dan kebutuhan akurasi segmentasi(Yu dkk., 2023).

2.1.5 *Segment Anything Model* (SAM)

Segment Anything Model (SAM) adalah model dasar (*foundation model*) pertama untuk segmentasi gambar umum yang diperkenalkan oleh Meta AI Research (Y. Wang dkk., 2024). Model ini dirancang sebagai model yang dapat diperintahkan (*promptable*), yang mampu melakukan segmentasi pada objek apa pun dalam citra atau video tanpa perlu pelatihan tambahan untuk objek baru tersebut, atau sebuah kemampuan yang dikenal sebagai *zero-shot transfer*. SAM dilatih menggunakan dataset SA-1B yang berskala sangat besar, yang berisi lebih dari 11 juta gambar dan 1 miliar *mask* segmentasi(Y. Wang dkk., 2024). Baru-baru ini, Meta juga merilis SAM 2, yang menyatukan kemampuan segmentasi gambar dan video dalam satu model dengan memori streaming untuk pemrosesan video secara *real-time* (Ravi dkk., 2024; C. Zhang dkk., 2023).

Fitur paling menonjol dari SAM adalah kemampuannya menerima berbagai jenis perintah (*prompt*) atau input interaktif dari pengguna, seperti titik, kotak pembatas (*Bounding Box*), atau teks untuk membantu area yang akan di segmentasi (Kirillov dkk., 2023; Y. Wang dkk., 2024). SAM dirancang untuk "sadar ambiguitas", yang berarti jika sebuah perintah tidak jelas model dapat menghasilkan beberapa prediksi segmentasi yang valid sekaligus (Kirillov dkk., 2023). Kemampuan generalisasi *zero-shot* memungkinkan SAM beradaptasi dengan distribusi gambar baru, seperti foto bawah tanpa pelatihan ulang (Kirillov dkk., 2023; C. Zhang dkk., 2023).



Gambar 2.8. Arsitektur *Segment Anything Model*

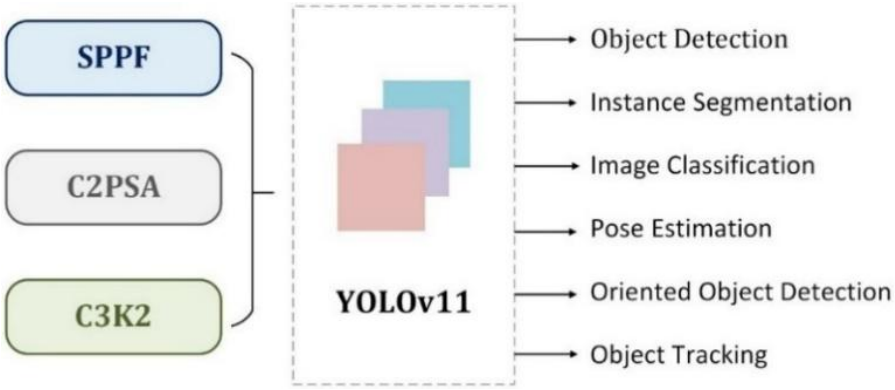
Gambar 2.8 memperlihatkan arsitektur SAM yang terdiri dari tiga komponen utama, yakni *image encoder*, *prompt encoder*, dan *mask decoder* yang ringan. Pada tahap *image encoder*, SAM menggunakan pendekatan *Vision Transformer* (VIT) sebagai encoder gambar. ViT ini merupakan sebuah jaringan saraf yang dirancang untuk memproses tambalan gambar seperti token di dalam NLP. *Image encoder* ini memproses gambar *input* satu kali untuk menghasilkan *embedding* gambar. Bersamaan dengan *image encoder, prompt encoder* juga melakukan pemetaan input gambar (perintah lokasi atau titik mana yang dijadikan rujukan) menjadi vektor representasi.

Informasi dari *image encoder* dan *prompt encoder* kemudian digabungkan ke dalam mask decoder untuk memprediksi area segmentasi secara efisien dan *real time*. Pada SAM 2, arsitektur SAM diperluas dengan mekanisme perhatian memory (*memory attention*) yang mengondisikan fitur frame saat ini berdasarkan frame dan prediksi masa lalu untuk menjaga konsistensi temporal dalam sebuah video.

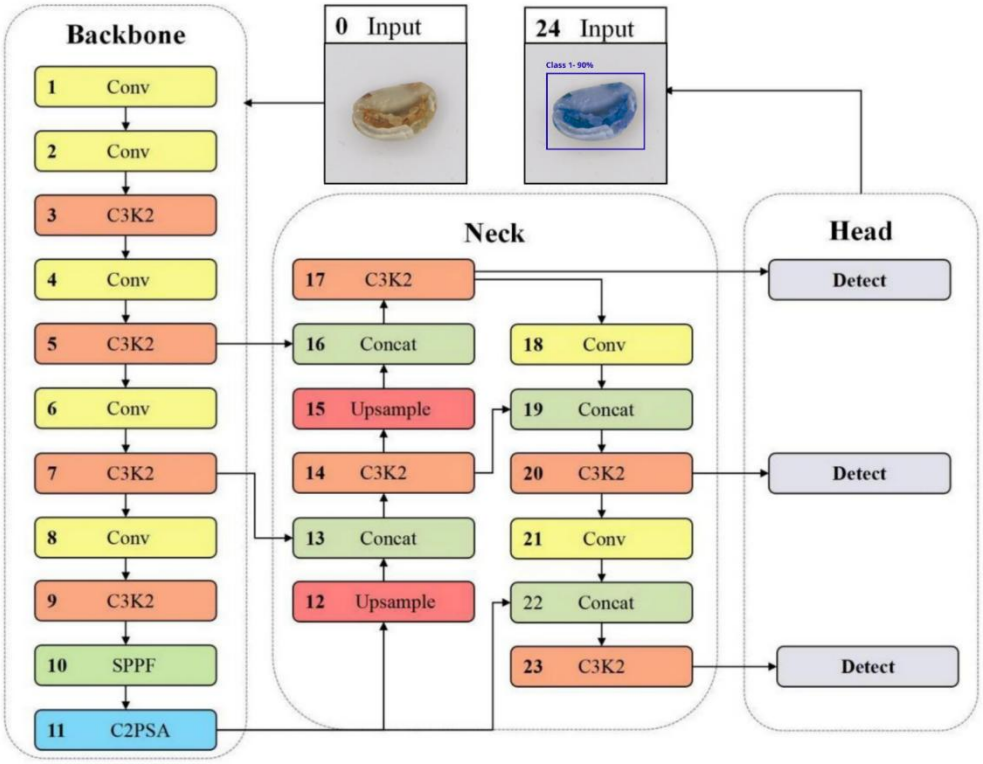
2.1.6 YOLOv11 (You Only Look Once Version 11)

YOLOv11 (*You Only Look Once Version 11*) merupakan algoritma terbaru dari seri algoritma deteksi objek keluarga YOLO yang dikembangkan oleh Ultralytics dan dipublikasikan pada September 2024. YOLOv11 dirancang untuk mendefinisikan ulang standar efisiensi dan akurasi dalam visi komputer secara real time. YOLOv11 dikategorikan sebagai model *single-stage detector*, di mana model ini tidak terbatas pada deteksi objek (YOLOv11), melainkan model ini dikategorikan sebagai model yang *multitask*, seperti segmentasi (YOLOv11-seg), klasifikasi gambar (YOLOv11-cls), estimasi pose (YOLOv11-pose), dan deteksi berorientasi (YOLOv11-obb) (Khanam & Hussain, 2024).

YOLOv11-seg digunakan sebagai model utama dalam deteksi objek berbasis segmentasi pada penelitian ini. YOLOv11-seg adalah model yang dikembangkan untuk model deksi yang mampu mengidentifikasi keberadaan objek dengan *Bounding Box* dan area segmentasi. YOLOv11-seg memiliki keunggulan yang terletak pada inovasi arsitekurnya, arsitektur YOLOv11-seg secara fundamental sama dengan YOLOv11 yang terdiri dari tiga komponen utama, yaitu *Backbone*, *Neck,* dan *Head* (Hidayatullah dkk., 2025; Khanam & Hussain, 2024). Arsitektur YOLOv11 terdapat pada Gambar 2.9 dan 2.10.



Gambar 2.9. Kunci Arsitektur YOLOv11 (Khanam & Hussain, 2024)



Gambar 2.10. Arsitektur YOLOv11

1. *Backbone*

   Backbone atau tulang punggungnya model merupakan komponen terpenting dalam arsitektur YOLO. Tugas dari komponen ini adalah bertanggung jawab untuk mengekstrak fitur dan citra input pada berbagai skala. Pembaruan yang terdapat pada YOLOv11 dibandingkan sebelumnya terletak pada blok dan

parameter yang digunakan. YOLOv11 menggunakan blok C3k2 (*Cross Stage Partial with Kernel Size 2*) untuk menggantikan blok C2f yang digunakan pada *backbone* versi sebelumnya.

2. *Neck*

   Neck pada YOLOv11 digunakan sebagai penghubung antara *backbone* dan *head* dengan tujuan untuk menggabungkan fitur dari berbagai skala. Pembaruan yang terdapat pada YOLOv11 dibandingkan versi sebelumnya adalah adanya C2PSA (*Convolutional block with Parallel Spatial Attention*), yang menjadi. *Blok* ini memungkinkan model untuk mempelajari hubungan spasial antara fitur secara lebih kontekstual melalui mekanisme *attention*. Mekanisme *attention* ini memungkinkan model untuk memusatkan perhatiannya pada wilayah yang ditargetkan dalam citranya, yang berpotensi menghasilkan deteksi yang lebih akurat terutama untuk objek yang lebih kecil atau terdapat gangguan.

3. *Head*

   Bagian *head* YOLOv11 berfungsi untuk menghasilkan keluaran akhir berupa *Bounding Box*, *skor objek*, dan kelas objek. *Head* menerima fitur multiskala dari *neck* dan memprosesnya melalui beberapa jalur deteksi untuk objek kecil, sedang, dan besar. YOLOv11 mengadopsi desain *head* yang mirip dengan YOLOv10, namun dengan optimasi tambahan seperti penggunaan *depthwise convolution* (DWConv) pada cabang klasifikasi.

2.1.7 Kuantisasi (*Quantization*)

Kuantisasi (*Quantization*) adalah salah satu teknik optimasi model yang bertujuan untuk mengurangi presisi numerik dari parameter model, seperti bobot

(*weights*) dan aktivasi (*activations*), dari representasi presisi yang tinggi misalnya FT32 ke representasi presisi lebih rendah seperti INT8 atau FP16 (Liu dkk., 2024). Tujuan utama dari *quantization* adalah untuk mengurangi ukuran model, kebutuhan memori, dan mempercepat proses inferensi, khususnya pada perangkat dengan sumber daya yang terbatas. Dengan menurunkan presisi numerik, operasi aritmatika dapat dijalankan menggunakan aritmatika integer yang lebih efisien secara komputasi, dengan penurunan akurasi yang relatif kecil jika dilakukan secara tepat (Hasanpour dkk., 2025).

Secara umum, *quantization* bekerja dengan memetakan nilai kontinu *floating-point* ke himpunan nilai diskret dalam rentang bit tertentu. Proses ini dilakukan menggunakan skema *uniform affine quantization*, di mana setiap nilai floating-point x_{int} melalui parameter skala (*scale*) dan titik nol (*zero-point*). Selama proses komputasi, nilai integer tersebut dikonversi kembali ke domain *floating-point* melalui proses *dequantization* (Hasanpour dkk., 2025). Berikut adalah persamaan 2.1 rumus *scale*, persamaan 2.2 rumus *zero-point,* persamaan 2.3 rumus *quantization* dan persamaan 2.4 rumus *dequantization* (Hasanpour dkk., 2025; Weng, 2021).

$$S = \frac{x_{max} - x_{min}}{q_{max} - q_{min}} \tag{2.1}$$

Pada Persamaan 2.1 nilai *scale* (*S*) merupakan faktor yang digunakan untuk memetakan rentang nilai asli (*floating-point*) ke rentang representasi integer. Nilai x_{max} dan x_{min} masing-masing menyatakan nilai terbesar dan terkecil dari data asli, sedangkan q_{max} dan q_{min} menunjukkan batas atas dan batas bawah dari rentang

integer yang digunakan, seperti 0 hingga 255 pada *unsigned integer* atau -128 hingga 127 pada *signed integer*.

$$Z = round\left(q_{min} - \frac{x_{min}}{S}\right) \tag{2.2}$$

Persamaan 2.2 digunakan untuk menghitung *zero-point* (*Z*), yaitu nilai integer yang memastikan angka nol pada domain *floating-point* dapat direpresentasikan secara tepat pada domain terkuantisasi. Nilai ini diperoleh dari hubungan antara batas bawah rentang integer (q_{min}), nilai minimum data (x_{min}), dan faktor *scale* (*S*). Fungsi *round* digunakan untuk membulatkan hasil perhitungan ke bilangan bulat terdekat.

$$q = clamp\left(round\left(\frac{x}{S} + Z\right), q_{min}, q_{max}\right) \tag{2.3}$$

Pada Persamaan 2.3, nilai asli *floating-point* (*x*) dikonversi menjadi nilai integer terkuantisasi (*q*). Proses ini dilakukan dengan membagi nilai asli menggunakan faktor scale (*S*), menambahkan *zero-point* (*Z*), kemudian membulatkannya ke bilangan bulat terdekat. Fungsi *clamp* digunakan untuk membatasi hasil kuantisasi agar tetap berada dalam rentang representasi integer yang ditentukan oleh q_{min} dan q_{max} .

$$d \approx S \times (q - Z) \tag{2.4}$$

Persamaan 2.4 menunjukkan proses dequantization, yaitu konversi kembali nilai integer terkuantisasi (*q*) ke bentuk *floating-point*. Proses ini dilakukan dengan mengurangi nilai integer terhadap *zero-point* (*Z*), kemudian mengalikannya dengan faktor skala (S). Hasil yang diperoleh merupakan estimasi dari nilai asli sebelum proses kuantisasi dilakukan.

Persamaan 2.1 sampai dengan 2.4 memiliki arti yang berbeda-beda. Faktor skala (*S*) berfungsi untuk memetakan rentang nilai riil ke rentang representasi integer target. Titik nol atau *zero-point* (*Z*) adalah parameter bertipe integer yang memastikan nilai riil 0.0 terpetakan secara eksak tanpa galat ke dalam domain terkuantisasi. Lalu, proses kuantisasi (*q*) merupakan proses konversi nilai desimal asli (x) ke bentuk integer terkuantisasi (*q*) yang dihitung dengan membagi nilai asli dengan skala, menambahkan titik nol, serta membatasi nilainya (*clamping*) agar tidak keluar dari rentang representasi target. Sedangkan *dequantization* (*d*) adalah proses untuk memulihkan beberapa akurasi yang hilang selama kuantisasi dengan cara memetakan angka *fixed-point* kembali ke angka *floating-point* selama pra-pemrosesan (Kumar, 2024; Liu dkk., 2025; Wu dkk., 2020).

Berdasarkan waktu penerapan, terdapat dua cara penggunaan *Quantization* yaitu *Post-Training Quantization* (PTQ) dan *Quantization-Aware Training* (QAT). PTQ adalah metode di mana model yang telah dilatih dengan presisi penuh (FP32) dikonversi menjadi presisi rendah tanpa pelatihan ulang, yang berguna ketika data pelatihan terbatas atau tidak berlabel. Sebaliknya, QAT menyisipkan simulasi kuantisasi selama proses pelatihan (*fine-tuning*), memungkinkan model untuk beradaptasi terhadap error atau noise yang timbul akibat penurunan presisi, sehingga biasanya menghasilkan akurasi yang lebih tinggi dibandingkan (Ahn dkk., 2023; Hasanpour dkk., 2025; Van Baalen dkk., 2023).

2.1.8 Matriks Akurasi Deteksi (*Performance Matrix*)

Evaluasi performa model bertujuan untuk mengetahui kemampuan model dalam mendeteksi dan mengklasifikasikan objek secara akurat. Pada penelitian ini,

performa model diukur menggunakan beberapa metrik evaluasi yang umum digunakan pada tugas deteksi dan segmentasi objek, seperti *Confusion Matrix*, *Precision*, *Recall* dan *Mean Average Precision* (mAP).

2.1.8.1. *Confusion Matrix*

Confusion matrix merupakan tabel yang membandingkan prediksi model dengan label asli (*ground-truth*). *Confusion matrix* berukuran $N \times N, di\ mana$ N adalah jumlah kelas keluaran. Setiap baris pada *confusion matrix* mewakili jumlah instance dari kelas prediksi dan setiap kolom mewakili jumlah instance dari kelas sebenarnya (Sathyanarayanan, 2024). Matriks ini memiliki dua variabel target yakni, positif dan negatif, yang terlihat pada Tabel 2.1.

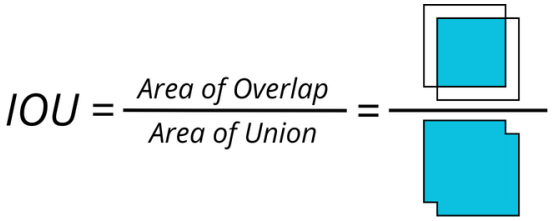
Tabel 2.1. *Confusion Matrix* (Sathyanarayanan, 2024)

		Actual Values	
		Positive	*Negative*
Predicted Values	*Positive*	*True Positive* (TP)	*False Positive* (FP)
	Negative	*False Negative* (FN)	*True Negative* (TN)

Berdasarkan Tabel 2.1 terdapat istilah TP, TN, FP, dan FN. *True Positive* (TP) menghitung jumlah benar yang memprediksi kelas positif di mana prediksi dan aktual keduanya dikatakan positif. *True Negatif* (TN) menghitung jumlah benar dalam memprediksi kelas negatif yang mana prediksi dan aktual keduanya negatif. *False Positive* (FP) menghitung jumlah prediksi yang salah dari kelas negatif yang di mana prediksi-nya positif, tapi nilai sebenarnya (aktual) adalah negatif. Terakhir *False Negative* (FN) menghitung jumlah prediksi yang salah pada kelas positif, yang mana prediksinya negatif tapi nilai sebenarnya (aktual) adalah positif (Sathyanarayanan, 2024).

2.1.8.2. *Intersection over Union* (IoU)

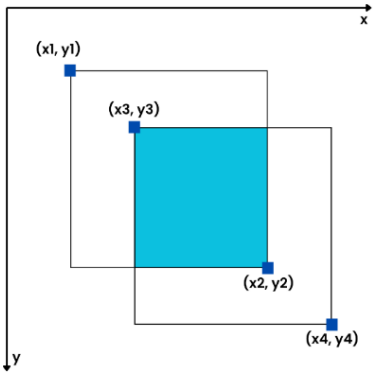
Intersection over Union (IoU) merupakan matriks fundamental yang mengukur rasio antara luas irisan dan luas gabungan antara *Bounding Box* atau segmentasi prediksi dan *ground-truth*, dengan ambang batas tertentu untuk mengklasifikasikan deteksi sebagai benar (1) atau salah (0) (Cho, 2024). Ilustrasi dan persamaan IoU terlihat pada Gambar 2.11 dan persamaan 2.5 (Padilla dkk., 2020).



Gambar 2.11. Illustrasi *Intersection over Union* (IoU)

$$J(B_p, B_{gt}) = IOU = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})} \quad (2.5)$$

Persamaan 2.5 secara komputasional, koordinat spasial kotak pembatas didefinisikan dalam format kartesian $[x_1, y_1, x_2, y_2]$ yang menunjukkan koordinat sudut kiri atas dan kanan bawah yang dapat dilihat pada Gambar 2.12.



Gambar 2.12. Visualisasi Cara Perhitungan IoU

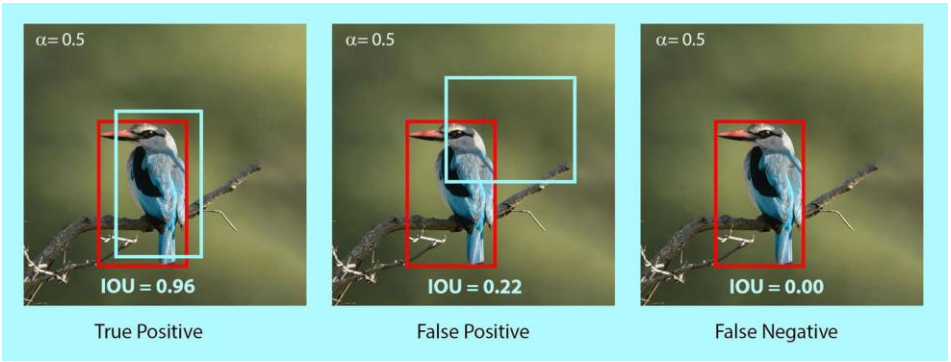
Berdasarkan Gambar 2.12, luas area irisan (*intersection area*) dihitung berdasarkan tumpang tindih koordinat, yang dirumuskan pada Persamaan 2.6 (Boesch, 2024).

$$\text{area}(B_p \cap B_{gt}) = (x_2 - x_3) \times (y_2 - y_3) \quad (2.6)$$

Sementara itu, luas area gabungan (*union area*) dihitung dengan menjumlahkan luas masing-masing kotak pembatas, kemudian dikurangi dengan luas area irisan untuk menghindari redundansi penghitungan piksel berganda. Berikut Persamaan 2.7 adalah rumus dari luas area gabungan.

$$\text{area}(B_p \cup B_{gt}) = |(x_2 - x_1) \times (y_2 - y_1)| + |(x_4 - x_3) \times (y_4 - y_3)| - |B_p \cap B_{gt}| \quad (2.7)$$

Adanya matriks IoU dapat membantu dalam memprediksi deteksi objek berdasarkan ambang batas dari IoU-nya sendiri. Ambang batas IoU ini akan memutuskan apakah prediksinya Benar Positif (TP), Positif Palsu (FP), atau Negatif Palsu (FN) . Contoh pada Gambar 2.12 menunjukkan prediksi dengan ambang batas (*threshold*) IoU yang ditetapkan pada 0,5.



a b c

Gambar 2.13. Contoh Pemanfaatan IoU dalam Prediksi Deteksi Objek (Kukil, 2022).

Pada Gambar 2.13 terdapat 3 prediksi yang masing-masing memiliki informasi kotak asli dan juga prediksi. Syarat untuk melakukan deteksi sebagai positif benar atau positif palsu sepenuhnya tergantung pada persyaratan yang digunakan. Sebagai contoh, ambang batas IoU yang digunakan adalah 0.5, maka pada prediksi Gambar 2.13a diprediksi sebagai positif benar dikarenakan nilai IoU sebesar 0.96 lebih besar dari pada ambang batas. Sedangkan pada prediksi kedua yang terdapat pada Gambar 2.13b diprediksi sebagai positif palsu karena ambang batas lebih besar dari pada nilai IoU. Prediksi ketiga dikategorikan sebagai negatif palsu karena nilai IoU = 0, yang berarti model gagal dalam mencari objek.

IoU dalam segmentasi merupakan matriks utama dalam mengevaluasi model. Pada segmentasi, perhitungan IoU tidak menggunakan kotak pembatas (*Bounding Box*) melainkan topeng dari segmentasi. Maka, faktor yang dijadikan sebagai perbanding adalah piksel demi piksel. Serta, definisi dari Benar Positif (TP), Positif Palsu (FP), atau Negatif Palsu (FN) berbeda dengan IoU *Bounding Box*, pada segmentasi TP, FP, dan FN adalah hasil dari prediksi area atau jumlah piksel (Huang dkk., 2020; Kukil, 2022).

Gambar 2.14. Visualisasi Prediksi IoU pada Segmentasi (Kukil, 2022).

Pada Gambar 2.14, area TP merupakan area persimpangan antara *ground truth* (GT) dengan topeng segmentasi (S). Area FP merupakan area yang diprediksi diluar GT. Sedangkan FN merupakan Jumlah piksel di area GT yang gagal di prediksi oleh model. Maka dari itu rumus perhitungan IoU untuk segmentasi terdapat pada persamaan 2.8 (Huang dkk., 2020; Kukil, 2022).

$$IoU_{seg} = \frac{TP}{(TP + FP + FN)} \tag{2.8}$$

Pada Persamaan 2.8, nilai *Intersection over Union* (IoU) menunjukkan tingkat kesesuaian antara hasil segmentasi yang diprediksi model dengan *ground truth*. Nilai TP merepresentasikan jumlah piksel yang berhasil diprediksi dengan benar sebagai objek, FP menunjukkan jumlah piksel yang diprediksi sebagai objek tetapi sebenarnya termasuk latar belakang, sedangkan FN menunjukkan jumlah piksel objek pada *ground truth* yang tidak berhasil dideteksi oleh model. Nilai IoU berada pada rentang 0 hingga 1, di mana nilai yang semakin mendekati 1 menunjukkan bahwa area hasil segmentasi memiliki tingkat tumpang tindih yang semakin tinggi dengan *ground truth*, sehingga mengindikasikan performa segmentasi yang semakin baik.

Keberadaan IoU atau mIoU pada deteksi objek tidak menghitung akurasi model secara langsung. Melainkan, matriks ini adalah metrik pembantu yang mengevaluasi tingkat tumpang tindih antara kebenaran dasar dan prediksi yang dapat digunakan oleh matriks akurasi lainnya, contohnya adalah mAP sebagai ambang batas penemuan lokasi objek sebelum prediksi TP, FP, dan FN (Kukil, 2022; Zhou & Jiang, 2025).

2.1.8.3. *Mean Average Precision* (mAP)

Mean Average Precision (mAP) adalah standar evaluasi pada deteksi objek. mAP merupakan perhitungan dari rata-rata *Average Precision* (AP) untuk seluruh kelas (C). AP sendiri merupakan rata-rata presisi yang di rumuskan pada persamaan 2.9, dan untuk rumus perhitungan mAP terdapat pada persamaan 2.10 (X. Zhang dkk., 2025).

$$AP = \int_0^1 p(r)dr \tag{2.9}$$

$$mAP = \frac{\sum_{i=1}^{C} AP}{C} \tag{2.10}$$

Pada Persamaan 2.9, nilai *Average Precision* (AP) diperoleh dari luas area di bawah kurva *Precision-Recall*. Fungsi $p(r)$ menyatakan nilai Precision sebagai fungsi dari *Recall* (r), dengan nilai *Recall* berada pada rentang 0 hingga 1. Semantara itu Persamaan 2.10 digunakan untuk menghitung *Mean Average Precision* (mAP), yaitu rata-rata nilai AP dari seluruh kelas objek yang dievaluasi. Nilai AP menunjukkan *Average Precision* pada kelas ke-i, sedangkan C menyatakan jumlah total kelas objek dalam dataset. Dengan menghitung rata-rata AP dari seluruh kelas, mAP dapat memberikan gambaran umum mengenai performa model dalam mendeteksi berbagai kategori objek secara keseluruhan.

Berdasarkan definisi dari IoU, IoU dapat digunakan dalam mAP sebagai pengaturan ambang batas pencarian lokasi objek (X. Zhang dkk., 2025). IoU dalam mAP biasanya terdapat pada matriks akurasi mAP50 dan juga mAP50-95. mAP50 akan menghasilkan deteksi yang benar selama tumpang tindih lebih dari ambang batas IoU 0.5 atau 50% dengan kotak sebenarnya (*ground truth*). Sedangkan untuk

mAP50-95 merupakan evaluasi dengan IoU multiambang, yakni kelipatan 5 dari 50 sampai dengan 95 atau 0.05 dari 0.5 sampai dengan 0.95. Angka 50 dan juga 50-95 merupakan ambang batas yang digunakan untuk evaluasi IoU. Jika area IoU lebih besar dari ambang batas, maka akan menghasilkan deteksi, dan deteksi tersebut akan dimasukkan ke dalam kategori TP, TN, atau FN yang mana akan dievaluasi oleh matriks mAP-nya sendiri (Padilla dkk., 2020; X. Zhang dkk., 2025).

Ketika ambang batas IoU diatur pada nilai yang lebih besar (seperti IoU 0,75 atau 0,95), toleransi terhadap pergeseran kotak menjadi sangat ketat. Evaluasi ketat membutuhkan tingkat tumpang tindih yang tinggi antara kotak prediksi dan kotak kebenaran sebenarnya. Sedangkan, penggunaan evaluasi multi-ambang (misalnya, IoU 0.5-0.95) akan menghitung rata-rata AP di 10 ambang berbeda dari 0,5 hingga 0,95, sehingga model menilai berbagai tingkat keketatan dan mengurangi bias akibat penggunaan ambang tunggal (X. Zhang dkk., 2025). Maka dari itu pilihan ambang IoU mewujudkan pertimbangan antara presisi dan *Recall*, yang secara langsung memengaruhi seberapa baik metrik mAP menangkap berbagai dimensi kemampuan model (Padilla dkk., 2020; X. Zhang dkk., 2025).

2.1.8.4. *Precision* (Presisi)

Precision atau presisi adalah perhitungan persentase yang diprediksi positif. Presisi ini bertujuan untuk mengukur seberapa akurat prediksi positif yang dihasilkan oleh model (Sathyanarayanan, 2024). Berikut formula mencari nilai persentase presisi yang terdapat pada persamaan 2.11 (Khan dkk., 2024; Sathyanarayanan, 2024).

$$Precision = \frac{TP}{TP\ +\ FP} \tag{2.11}$$

Pada Persamaan 2.11, nilai *Precision* dihitung sebagai perbandingan antara jumlah *True Positive* (TP) dan total prediksi positif yang dihasilkan model, yaitu jumlah *True Positive* (TP) ditambah *False Positive* (FP). Nilai TP menunjukkan jumlah objek yang berhasil diprediksi dengan benar sebagai kelas positif, sedangkan FP menunjukkan jumlah objek yang diprediksi sebagai kelas positif tetapi sebenarnya termasuk kelas lain atau bukan objek yang dimaksud (Sathyanarayanan, 2024).

2.1.8.5. *Recall*

Recall adalah metrik yang menunjukkan seberapa banyak data positif (objek asli) yang berhasil ditemukan kembali oleh model. Metrik ini berguna untuk mengukur kelengkapan deteksi, terutama ketika kita ingin meminimalkan *false negatives* (Sathyanarayanan, 2024). Persamaan 2.12 merupakan rumus perhitungan dari *Recall* (Khan dkk., 2024; Sathyanarayanan, 2024).

$$Recall = \frac{TP}{TP\ +\ FN} \tag{2.12}$$

Pada Persamaan 2.12, nilai *Recall* dihitung sebagai perbandingan antara jumlah *True Positive* (TP) dan total data positif sebenarnya, yaitu jumlah *True Positive* (TP) ditambah *False Negative* (FN).

2.1.8.6. *F1-Score*

F1-Score adalah rata-rata gabungan dari *Precision* dan *Recall*. Metrik ini memberikan gambaran keseimbangan performa, terutama ketika terdapat ketidakseimbangan (*imbalance*) antara *Precision* dan *Recall*. Persamaan 2.13 merupakan rumus perhitungan dari *F1-Score* (Khan dkk., 2024; Sathyanarayanan, 2024).

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.13)$$

2.1.8.7. *Frames Per Second* (FPS)

Frames Per Second (FPS) merupakan pengukuran kecepatan pemrosesan model dalam satuan gambar per detik. FPS digunakan untuk melihat seberapa cepat model dalam mendeteksi gambar. Semakin tinggi FPS, semakin cepat model mendeteksi objek. Serta FPS merupakan perbandingan dari latensi yang di mana lateni merupakan waktu total proses untuk satu gambar dalam milidetik (Verdianto, 2023).

2.1.8.8. Ukuran

Ukuran model merupakan pengukuran kapasitas memori yang dibutuhkan untuk menyimpan parameter jaringan saraf tiruan (weights), yang biasanya dinyatakan dalam satuan byte atau megabyte (MB) . Ukuran model digunakan untuk melihat seberapa efisien model tersebut dalam penggunaan ruang penyimpanan, terutama untuk penerapan pada perangkat dengan sumber daya terbatas (X. Zhang dkk., 2025). Ukuran model merupakan hasil perkalian dari total jumlah parameter dengan ukuran byte per parameter yang ditentukan oleh presisi bilangan, contohnya 4 byte untuk FP32, 2 byte untuk FP16, 1 byte untuk INT8, dan 0,5 byte untuk INT4. Berikut persamaan 2.14 merupakan rumus dari perhitungan estimasi ukuran model (Naminas, 2024).

$$Ukuran\ Model = Total\ Paramter \times Byte\ per - Parameter \quad (2.14)$$

2.2 *State of The Art*

Berbagai peneliti telah memanfaatkan metode deep learning untuk klasifikasi kualitas dan deteksi cacat pada biji kopi menggunakan beragam pendekatan, seperti *transfer learning*, *convolutional neural network*, *transformer*, serta *object detection* berbasis YOLO. Penelitian-penelitian tersebut menggunakan objek dan skenario yang beragam, mulai dari pengembangan dataset, komparasi arsitektur model, hingga implementasi sistem pada berbagai perangkat dan aplikasi *real-time*. Perbedaan objek penelitian, algoritma yang digunakan, serta hasil menunjukkan terdapat keragaman pendekatan dalam menyelesaikan permasalahan klasifikasi dan deteksi biji kopi. Berikut Tabel 2.2 adalah ringkasan penelitian-penelitian terkait yang menjadi rujukan dalam penelitian ini.

Tabel 2.2. *State of The Art*

No	***State of The Art***	
1	Peneliti, Tahun	(Arwatchananukul dkk., 2024)
	Judul	*Implementing a deep learning model for defect classification in Thai Arabica green coffee beans*
	Objek	Pengembangan model CNN untuk klasifikasi 17 jenis cacat biji kopi hijau
	Algoritma	MobileNetV3 (Best Model), MobileNetV2, EfficientNetV2, InceptionV2, dan ResNetV2, *Quantization* & Pruning.
	Hasil	**Hasil:** akurasi model MobileNetV3 = 90,19%, kuantisasi model 5,63 MB. **Kekurangan:** misklasifikasi pada cacat yang memiliki kemiripan visual. **Rekomendasi:** *ensemble learning* dan penerapan *augmentasi* lanjutan.

No	*State of The Art*	
2	Peneliti, Tahun	(Ekowicaksono dkk., 2025)
	Judul	*Transformer-Based Deep Learning Model for Coffee Bean Classification*
	Objek	Pelatihan dan evaluasi model Deep Learning berbasis *Transformer* untuk klasifikasi citra biji kopi Arabica
	Algoritma	*Vision Transformer* (ViT) dan *Swin Transformer*
	Hasil	**Hasil:** akurasi model *Swin Transformer* = 95,13%. **Kekurangan:** sumber daya komputasi yang signifikan. **Rekomendasi:** fokus pada optimasi model yang dapat diimplementasikan pada perangkat *low-end*.
3	Peneliti, Tahun	(Febriana dkk., 2022)
	Judul	*USK-COFFEE Dataset: A Multi-Class Green Arabica Coffee Bean Dataset for Deep Learning*
	Objek	Pembuatan dataset USK-Coffee multi-kelas kopi Arabika hijau.
	Algoritma	*Convolutional Neural Network* (CNN), MobileNetV2, ResNet-18
	Hasil	**Hasil:** akurasi model MobileNetV2 : 81,31%. **Kekurangan:** kesalahan klasifikasi. **Rekomendasi:** penambahan variasi data.
4	Peneliti, Tahun	(Hashmi dkk., 2025)
	Judul	*VGGBM-Net: A Novel Pixel-Based Transfer Features Engineering for Automated Coffee Bean Diseases Classification*
	Objek	Usulan model VGGBM-Net berbasis *transfer learning* untuk klasifikasi penyakit atau cacat biji kopi.
	Algoritma	VGGBM-Net (Ekstraksi fitur VGG-19 dikombinasikan dengan LightGBM (LGBM) *classifier*.)

No	*State of The Art*	
	Hasil	**Hasil:** akurasi model LightGBM (LGBM) 99%. **Kekurangan:** skalabilitas komputasi pada aplikasi *real-time*. **Rekomendasi:** model yang cocok untuk *drone* atau sistem *mobile* dan penerapan optimasi model.
5	Peneliti, Tahun	(Thai dkk., 2024)
	Judul	*Coffee Bean Defects Automatic Classification Realtime Application Adopting Deep Learning*
	Objek	Implementasi *Deep Learning* dalam aplikasi mobile untuk klasifikasi kualitas dan cacat biji kopi.
	Algoritma	YOLOv8
	Hasil	**Hasil:** kecepatan model mendeteksi 1 hingga 3 detik per inferensi. **Kekurangan:** *F1-score* 38%. **Rekomendasi:** perluasan dataset, penyempurnaan *hyperparameter* model.
6	Peneliti, Tahun	(Muchtar dkk., 2025)
	Judul	*Edge AI-Based Detection for Defective Coffee Beans Using Deep Learning and Streamlit Framework*
	Objek	Deteksi biji kopi cacat dengan komparasi performa model CNN vs *Transformer*
	Algoritma	Edge AI (NVIDIA Jetson Orin Nano) dengan model: FocalNet, ResNet-34, VGG-16, EfficientNet-B7, DenseNet121, InceptionV4, ViT, Swin *Transformer*, ConvNeXt,
	Hasil	**Hasil:** akurasi model FocalNet 93% dan *F1-score* 93%. **Kekurangan:** evaluasi model dilakukan pada satu jenis perangkat. **Rekomendasi:** eksperimen menggunakan berbagai jenis perangkat lain.
7	Peneliti, Tahun	(Oncu, 2025)
	Judul	*Deep Learning Framework for Coffee Quality Assessment via YOLOv8n Object Detection of Bean Defects*

No	*State of The Art*	
	Objek	Deteksi objek biji kopi cacat menggunakan YOLOv8n.
	Algoritma	YOLOv8n & Grad-CAM
	Hasil	**Hasil:** MCC model YOLOv8n dengan Grad-CAM 73%. **Kekurangan:** kurang data set. **Rekomendasi:** perluasan data melalui *augmentasi* dan pemanfaatan data *hiperspektral.*
8	Peneliti, Tahun	(Chang & Liu, 2024)
	Judul	*Multiscale Defect Extraction Neural Network for Green Coffee Bean Defects Detection*
	Objek	Usulan model *Deep Learning* ekstraksi cacat multiskala (*multiscale defect extraction*) untuk mendeteksi berbagai jenis cacat pada biji kopi hijau.
	Algoritma	*Multiscale Defect-Detection Network*
	Hasil	**Hasil:** akurasi *Multiscale Defect Extraction* (MDE) 96%. **Kekurangan:** penggunaan kernel konvolusi yang digunakan menghilangkan fitur penting. **Rekomendasi:** penyesuaian skala dan optimasi arsitektur.
9	Peneliti, Tahun	(Gope dkk., 2024)
	Judul	*Comparative analysis of YOLO models for green coffee bean detection and defect classification*
	Objek	Perbandingan model YOLO (v3-v8 & *Custom*) untuk deteksi dan klasifikasi biji kopi hijau beserta cacatnya
	Algoritma	YOLOV3, YOLOv4, YOLOV5, YOLOv7, YOLOv8, *Custom*-YOLOv8n
	Hasil	**Hasil:** mAP model YOLOv8n 0,995. **Kekurangan:** proses pelatihan memerlukan waktu yang lama. **Rekomendasi:** perluasan dataset, dan pengembangan pada perangkat *low-end*.

No	*State of The Art*	
10	Peneliti, Tahun	(Arfan Astaraja dkk., 2025)
	Judul	*Optimizing Coffee Ripeness Classification Using Yolov5 for Automated Detection and Sorting*
	Objek	Pengembangan sistem deteksi klasifikasi biji kopi (matang, tidak matang, dan busuk) secara *real-time*.
	Algoritma	YOLOv5
	Hasil	**Hasil:** akurasi YOLOv5 dengan Raspberry Pi 90%. **Kekurangan:** sistem terlalu efisien dan memiliki tingkat kehilangan model yang rendah. **Rekomendasi:** peningkatan akurasi lebih lanjut, penggunaan dataset yang lebih besar, dan optimasi algoritma.
11	Peneliti, Tahun	(He, Zhou, Liu, Zhang, dkk., 2025)
	Judul	*Application of the YOLOv11-seg Algorithm for AI-Based Landslide Detection and Recognition*
	Objek	Deteksi dan segmentasi area longsor berbasis citra menggunakan dataset Bijie-Landslide
	Algoritma	YOLOv11-seg
	Hasil	**Hasil:** *F1-score* model YOLOv11-seg 87,81%. **Kekurangan:** sumber daya komputasi yang relatif tinggi. **Rekomendasi:** mengembangkan model yang lebih ringan, mengoptimalkan arsitektur untuk edge *computing*.
12	Peneliti, Tahun	(Zhong dkk., 2025)
	Judul	*A Classification Method for the Severity of Aloe Anthracnose Based on the Improved YOLOv11-seg*
	Objek	Deteksi dan klasifikasi tingkat keparahan penyakit daun pada tanaman lidah buaya.
	Algoritma	YOLOv11-seg-DEDB

No	*State of The Art*	
	Hasil	**Hasil:** peningkatan akurasi model sebesar 5,3% dan mAP50 3,4%. **Kekurangan:** pembuatan anotasi manual menggunakan LabelMe. **Rekomendasi:** integrasi YOLOv11-seg dengan sistem monitoring berbasis *real-time*.
13	Peneliti, Tahun	(Cao dkk., 2025)
	Judul	*Automatic Pipeline Defect Detection Based on An Improved YOLOV11-seg Instance Segmentation Network*
	Objek	Deteksi cacat saluran pipa limbah bawah tanah menggunakan teknik segmentasi instan.
	Algoritma	YOLOv11-seg, BiPANet, ResSPP
	Hasil	**Hasil:** mAP model YOLOv11-seg 87,2%. **Kekurangan:** *missed detection* dan *false detection* pada lingkungan yang redup. **Rekomendasi:** pengembangan *lightweight deployment*.
14	PenelitiTahun	(He dkk., 2024)
	Judul	*Research and Application of YOLOv11-Based Object Segmentation in Intelligent Recognition at Construction Sites*
	Objek	Deteksi dan segmentasi multi-objek pada situs konstruksi cerdas, mencakup 13 kategori target.
	Algoritma	YOLOv11-seg
	Hasil	**Hasil:** mAP@0.5 model YOLOv11-Seg 80.8%. **Kekurangan:** pelabelan anotasi secara manual menggunakan LabelMe. **Rekomendasi:** untuk memperluas cakupan deteksi ke kategori lain.

Berdasarkan Tabel 2.2 yang berisikan rangkuman berbagai penelitian terdahulu, menunjukkan bahwa pendekatan deep learning telah banyak diterapkan untuk deteksi, klasifikasi, dan penilaian kualitas biji kopi menggunakan model CNN dan *Transformer*. Penelitian-penelitian tersebut menghasilkan capaian akurasi yang tinggi, mulai dari 90% hingga di atas 99%, baik pada klasifikasi cacat biji kopi, deteksi kualitas, maupun segmentasi objek pada berbagai objek, termasuk pertanian dan inspeksi visual. Namun, sebagian besar penelitian masih menghadapi keterbatasan berupa misklasifikasi pada cacat dengan kemiripan visual, kebutuhan sumber daya komputasi yang tinggi, keterbatasan generalisasi akibat ukuran dan variasi dataset, serta teknik anotasi yang masih manual menggunakan *software* tertentu. Mengacu pada temuan tersebut, penelitian akan memanfaatkan model YOLOv11 untuk deteksi cacat biji kopi berbasis segmentasi yang dioptimalkan dengan teknik *quantization*, dengan tujuan menjaga keseimbangan antara akurasi, efisiensi komputasi, dan kemampuan implementasi pada perangkat *low-end*. Serta pemanfaatan *Segment Anything Model* sebagai sistem anotasi otomatis untuk area segmentasi biji kopi.

2.3 Matriks Penelitian

Tabel 2.3. Matriks Penelitian

No	Penulis	Model			Anotasi			Deteksi			Kuantisasi	Evaluasi Model			Implementasi Real Time
		CNN	YOLO	*Transformer*	Manual	Semi Otomatis	Otomatis	Klasifikasi	Objek	Segmentasi		Akurasi	Ukuran	Kecepatan	
1	(Arwatchananukul dkk., 2024)	✓	-	-	-	-	-	✓	-	-	✓	✓	✓	✓	✓
2	(Ekowicaksono dkk., 2025)	-	-	✓	-	-	-	✓	-	-	-	✓	✓	-	-
3	(Febriana dkk., 2022)	✓	-	-	-	-	-	✓	-	-	-	✓	-	✓	-
4	(Hashmi dkk., 2025)	✓	-	-	-	-	-	✓	-	-	-	✓	-	✓	-
5	(Thai dkk., 2024)	-	✓	-	-	-	-	✓	✓	-	-	✓	-	✓	✓
6	(Muchtar dkk., 2025)	✓	-	✓	-	-	-	✓	-	-	-	✓	-	✓	✓
7	(Oncu, 2025)	-	✓	-	-	-	-	✓	✓	-	-	✓	-	-	✓
8	(Chang & Liu, 2024)	✓	-	-	-	-	-	✓	-	-	-	✓	✓	-	-
9	(Gope dkk., 2024)	-	✓	-	-	-	-	✓	✓	-	-	✓	-	-	✓
10	(Arfan Astaraja dkk., 2025)	-	✓	-	-	-	-	✓	✓	-	-	✓	-	-	✓
11	(He, Zhou, Liu, Zhang, dkk., 2025)	-	✓	-	✓	-	-	-	✓	✓	-	✓	-	-	✓
12	(Zhong dkk., 2025)	-	✓	-	✓	-	-	-	✓	✓	-	✓	✓	✓	-
13	(Cao dkk., 2025)	-	✓	-	-	-	-	-	✓	✓	-	✓	-	✓	✓
14	(He dkk., 2024)	-	✓	-	✓	-	-	-	✓	✓	-	✓	-	-	✓
15	Usulan Penelitian	-	✓	-	-	-	✓	✓	✓	✓	✓	✓	✓	✓	✓

Berdasarkan Tabel 2.3, mayoritas penelitian terdahulu seperti yang dilakukan oleh (Arwatchananukul dkk., 2024) dan (Hashmi dkk., 2025) berfokus pada klasifikasi menggunakan pendekatan CNN yang digunakan untuk klasifikasi. Serta penelitian yang dilakukan oleh (Arfan Astaraja dkk., 2025; Gope dkk., 2024; Thai dkk., 2024) menggunakan pendekatan deteksi objek berbasis YOLO, dengan hasil penelitian yang unggul dalam kecepatan *real-time* namun terbatas pada *Bounding Box* yang kurang presisi menangkap morfologi cacat biji kopi. Sementara terdapat juga penelitian yang menggunakan pendekatan *Transformer* dengan hasil akurasi yang tinggi dalam menangkap konteks global, namun memiliki keterbatasan dalam komputasi yang terlalu berat untuk diterapkan secara efesien pada perangkat *edge* dengan sumber daya terbatas (Ekowicaksono dkk., 2025; Muchtar dkk., 2025),

Dari penelitian terdahulu, terdapat kesenjangan dalam menyeimbangkan kebutuhan anotasi yang presisi untuk segmentasi citra dengan efisiensi komputasi pada perangkat *low-end* yang belum sepenuhnya teratasi oleh metode deteksi *Bounding Box*. Maka dari itu penelitian ini mengusulkan penggunaan arsitektur YOLOv11 yang dikombinasikan dengan teknik segmentasi dan *quantization*, bertujuan untuk menghasilkan model deteksi objek yang akurat secara visual namun tetap ringan (*lightweight*), sehingga mampu beroperasi secara *real-time* pada perangkat *low-end*.