BAB 2
LANDASAN TEORI

Pengembangan sistem didukung oleh berbagai konsep dan metode. Oleh karena itu, bab ini menguraikan landasan teori yang meliputi tablet paracetamol, citra digital, computer vision, deep learning, object detection, arsitektur YOLOv8, teknik validasi model, metrik evaluasi, serta teknologi pendukung.

2.0.1 Penelitian Terdahulu

Berbagai penelitian telah memanfaatkan metode *deep learning* untuk mendukung identifikasi pil maupun inspeksi kualitas tablet. Penelitian-penelitian tersebut memiliki fokus yang beragam, seperti deteksi cacat tablet pada proses manufaktur, identifikasi berbagai jenis pil, maupun pengembangan sistem untuk kebutuhan pengguna tertentu. Meskipun demikian, belum banyak penelitian yang secara khusus menerapkan YOLOv8 untuk mengidentifikasi tablet paracetamol serta mengintegrasikannya ke dalam aplikasi berbasis web. Ringkasan penelitian terdahulu dan identifikasi *research gap* disajikan pada Tabel 2.1.

Tabel 2.1. *Research Gap* Penelitian Terdahulu

Penelitian	**Fokus**	***Research Gap***	**Kontribusi**
Kwon et al. (2021) [6]	Identifikasi pil berbasis *Mask R-CNN*.	Belum menggunakan YOLOv8 dan variasi citra terbatas.	YOLOv8 untuk identifikasi tablet paracetamol pada berbagai kondisi citra.
Heo et al. (2023) [7]	Identifikasi berbagai jenis pil.	Tidak berfokus pada tablet paracetamol.	Identifikasi tablet paracetamol menggunakan YOLOv8.
Dang et al. (2024) [8]	Aplikasi identifikasi pil berbasis YOLO.	Berorientasi pada aksesibilitas pengguna.	Sistem berbasis web dengan evaluasi performa dan *usability*.
Vijayakumar et al. (2024) [11]	Deteksi cacat tablet menggunakan YOLOv8.	Difokuskan pada inspeksi kualitas tablet.	Identifikasi jenis tablet paracetamol.
Kavitha et al. (2025) [9]	Identifikasi pil menggunakan YOLOv5 dan OCR.	Menggunakan YOLOv5 dan OCR.	YOLOv8 berbasis karakteristik visual tablet.
Freiermuth et al. (2025) [12]	Deteksi cacat permukaan tablet.	Berfokus pada inspeksi kualitas tablet.	Identifikasi jenis tablet paracetamol.

Berdasarkan penelitian-penelitian terdahulu, dapat diketahui bahwa penerapan *deep learning* pada bidang farmasi telah banyak digunakan untuk inspeksi kualitas tablet maupun identifikasi berbagai jenis pil. Namun,

implementasi YOLOv8 yang secara khusus digunakan untuk mengidentifikasi tablet paracetamol dan diintegrasikan ke dalam aplikasi berbasis web masih belum banyak dikaji. Oleh karena itu, dikembangkan sistem identifikasi tablet paracetamol berbasis web menggunakan YOLOv8 yang dievaluasi menggunakan metrik *accuracy*, *precision*, *recall*, *F1-score*, serta *System Usability Scale* (SUS).

2.1 Obat Paracetamol

Paracetamol atau *acetaminophen* merupakan obat yang banyak digunakan untuk meredakan nyeri (*analgesik*) dan menurunkan demam (*antipiretik*), serta menjadi salah satu terapi utama untuk nyeri ringan hingga sedang [13, 3]. Obat ini bekerja dengan menghambat aktivitas enzim *cyclooxygenase* (COX) di sistem saraf pusat sehingga mengurangi pembentukan *prostaglandin* yang berperan dalam timbulnya nyeri dan demam [14].

Paracetamol tersedia dalam beberapa bentuk sediaan, yaitu tablet, kapsul, sirup, dan *suppositoria*. Sediaan tablet dipilih karena merupakan bentuk yang paling umum digunakan dan memiliki distribusi yang luas di masyarakat [13]. Meskipun memiliki kandungan zat aktif yang sama, setiap produsen memiliki karakteristik visual tablet yang berbeda, seperti bentuk, warna, ukuran, *imprint*, dan *score line*. Karakteristik tersebut dimanfaatkan sebagai fitur visual dalam proses *object detection*. Karakteristik visual tablet paracetamol ditunjukkan pada Tabel 2.2.

Tabel 2.2. Karakteristik Visual Tablet Paracetamol

Karakteristik	**Deskripsi**	**Pengaruh terhadap Deteksi**
Bentuk	Bulat, oval, atau lonjong.	Membedakan bentuk tablet.
Warna	Putih hingga kekuningan.	Membedakan tablet yang serupa.
Ukuran	Bervariasi menurut dosis dan merek.	Membedakan ukuran tablet.
Imprint	Tulisan, angka, atau logo.	Mengidentifikasi merek tablet.
Score Line	Garis pembagi tablet.	Menambah ciri visual.

Karakteristik visual tablet direpresentasikan dalam bentuk citra digital sehingga dapat diproses menggunakan metode *computer vision*. Konsep citra digital menjadi salah satu dasar dalam proses identifikasi objek menggunakan metode

object detection.

2.2 Citra Digital

Citra digital merupakan representasi visual suatu objek dalam bentuk data digital yang tersusun atas kumpulan piksel pada matriks dua dimensi sehingga dapat diproses oleh komputer. Setiap piksel menyimpan nilai intensitas cahaya, sedangkan kualitas citra dipengaruhi oleh resolusi spasial dan kedalaman bit (*bit depth*) yang menentukan banyaknya tingkat intensitas yang dapat direpresentasikan [15].

Dalam *computer vision*, dua representasi citra yang umum digunakan adalah citra RGB dan citra *grayscale*.

1. Citra RGB
   Citra RGB merepresentasikan warna menggunakan tiga saluran, yaitu merah (*Red*), hijau (*Green*), dan biru (*Blue*). Setiap saluran memiliki nilai intensitas 0–255 pada representasi 8-bit sehingga mampu menghasilkan berbagai variasi warna. Representasi ini sesuai untuk mempertahankan informasi visual, seperti perbedaan warna pada tablet paracetamol [15].

2. Citra Grayscale
   Citra *grayscale* hanya memiliki satu saluran intensitas dengan rentang nilai 0 (hitam) hingga 255 (putih). Citra ini diperoleh melalui konversi dari citra RGB menggunakan persamaan berikut.

$$Y = 0.299R + 0.587G + 0.114B \tag{2.1}$$

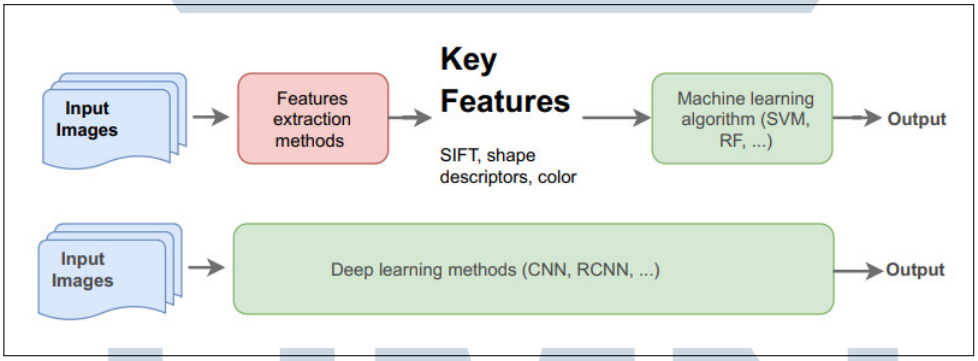
   dengan R, G, dan B masing-masing menyatakan intensitas saluran merah, hijau, dan biru. Meskipun lebih ringan secara komputasi, citra *grayscale* tidak mempertahankan informasi warna yang dapat menjadi fitur penting dalam proses identifikasi objek [15].

Pada tugas *object detection*, YOLOv8 menggunakan citra RGB sebagai masukan untuk mempertahankan informasi warna yang berperan dalam proses ekstraksi fitur dan identifikasi objek. Representasi citra digital tersebut menjadi dasar bagi metode *computer vision* dalam melakukan analisis dan pengenalan objek secara otomatis.

2.3 *Computer Vision*

Computer vision adalah salah satu cabang kecerdasan buatan yang memungkinkan komputer memahami dan menginterpretasikan informasi visual dari citra maupun video. Tujuan utamanya adalah mengenali objek, pola, serta informasi penting lainnya secara otomatis sehingga dapat dimanfaatkan pada berbagai aplikasi berbasis kecerdasan buatan [16, 17].

Perkembangan *computer vision* semakin pesat seiring kemajuan metode *deep learning*. Arsitektur seperti *Convolutional Neural Network* (CNN) dan *Vision Transformer* dapat mempelajari representasi visual secara otomatis sehingga memberikan peningkatan performa pada berbagai tugas, seperti klasifikasi citra, deteksi objek, dan segmentasi [16, 18]. Perbandingan antara metode *computer vision* tradisional dan pendekatan berbasis *deep learning* dapat dilihat pada Gambar 2.1.



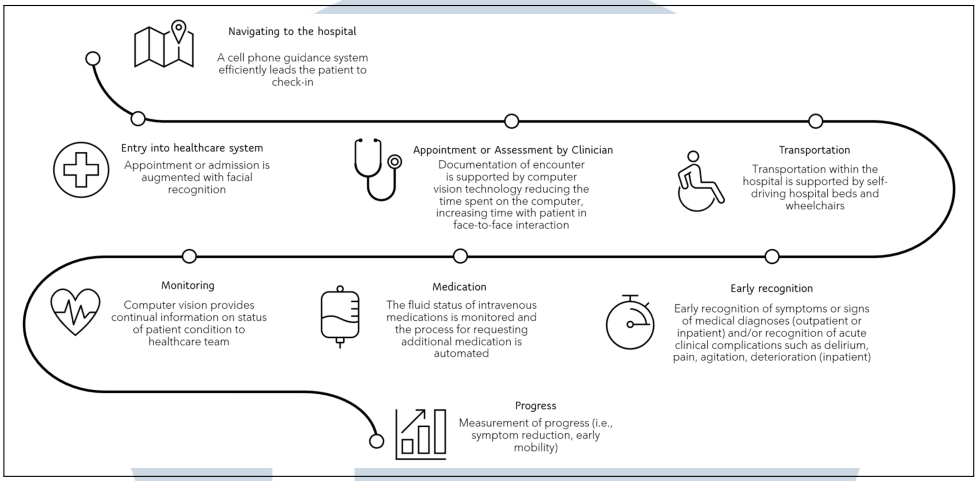
Gambar 2.1. Metode Tradisional Computer Vision (Atas) vs Pendekatan Deep Learning (Bawah)

Sumber: [16]

Secara umum, sistem *computer vision* meliputi empat langkah utama, yaitu akuisisi citra, *preprocessing*, ekstraksi fitur, serta klasifikasi atau deteksi objek. Tahapan tersebut memungkinkan citra diproses untuk memperoleh karakteristik visual, seperti bentuk, tekstur, pola, dan tepi objek. Pada pendekatan berbasis CNN, proses ekstraksi fitur dilakukan secara otomatis melalui lapisan konvolusi, sedangkan tahap akhir digunakan untuk menentukan keberadaan, lokasi, dan kategori objek dalam citra [19, 18, 20].

Penerapan *computer vision* telah berkembang pada berbagai bidang, khususnya kesehatan dan farmasi, seperti analisis citra radiologi, deteksi tumor, segmentasi organ, inspeksi kualitas obat, serta identifikasi jenis obat berbasis

citra [19, 17]. Beberapa contoh implementasi *computer vision* dalam pelayanan kesehatan ditunjukkan pada Gambar 2.2.

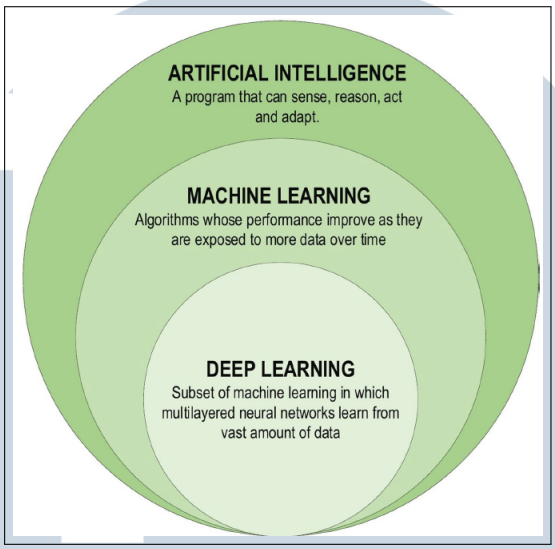


Gambar 2.2. Penerapan *computer vision* dalam sistem pelayanan kesehatan
Sumber: [19]

Kemampuan *computer vision* dalam mengenali pola visual secara otomatis didukung oleh perkembangan metode *deep learning*. Tidak seperti pendekatan tradisional yang bergantung pada perancangan fitur secara manual, *deep learning* mampu mempelajari representasi fitur langsung dari data, sehingga menjadi pendekatan yang banyak digunakan dalam berbagai tugas analisis citra, termasuk *object detection*.

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

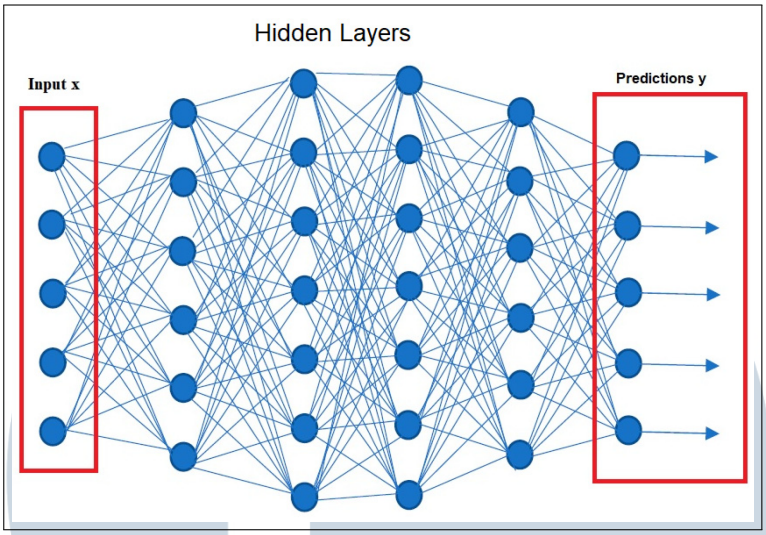
2.4 *Deep Learning*



Gambar 2.3. Hierarki AI, ML, dan DL
Sumber: [21]

Deep learning merupakan subbidang dari *machine learning* yang menggunakan jaringan saraf tiruan (*Artificial Neural Network*/ANN) dengan banyak lapisan (*hidden layers*) untuk mempelajari representasi data secara bertahap [21]. Sebagaimana ditunjukkan pada Gambar 2.3, *deep learning* merupakan bagian dari *machine learning*, sedangkan *machine learning* merupakan bagian dari *Artificial Intelligence* (AI). Berbeda dengan pendekatan konvensional yang memerlukan ekstraksi fitur secara manual, *deep learning* mampu mempelajari karakteristik data secara otomatis sehingga efektif dalam menangani data tidak terstruktur, seperti citra, audio, dan teks [22].

Dalam pengolahan citra, *deep learning* banyak digunakan pada berbagai tugas, seperti klasifikasi citra, deteksi objek, dan segmentasi karena mampu menangkap hubungan non-linear yang kompleks serta menghasilkan performa yang lebih baik dibandingkan metode konvensional [23]. Dasar dari pendekatan ini adalah ANN yang terinspirasi dari cara kerja neuron biologis dan tersusun atas neuron-neuron buatan yang saling terhubung melalui bobot (*weights*) yang diperbarui selama proses pelatihan [21].



Gambar 2.4. Arsitektur dari Neural Network

Sumber: [22]

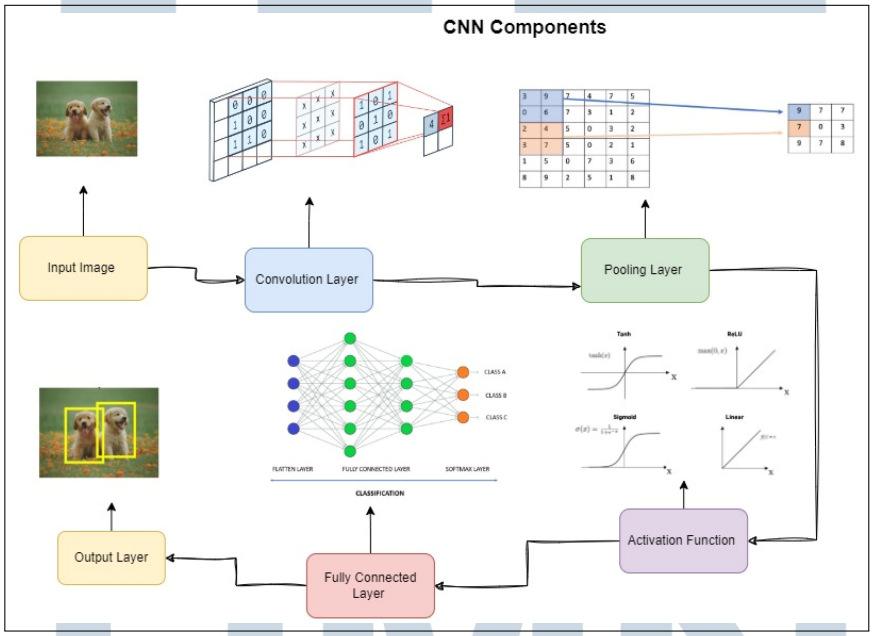
Berdasarkan Gambar 2.4, ANN terdiri atas tiga komponen utama, yaitu *input layer* sebagai penerima data masukan, *hidden layer* sebagai tempat proses pembelajaran dan ekstraksi fitur, serta *output layer* yang menghasilkan prediksi akhir. Selama proses pelatihan, bobot antar neuron diperbarui menggunakan algoritma *backpropagation* dengan metode optimasi, seperti *Stochastic Gradient Descent* (SGD) atau *Adam optimizer*. Selain itu, fungsi aktivasi, seperti ReLU, *Sigmoid*, dan *Softmax*, digunakan untuk memberikan sifat non-linear sehingga jaringan mampu mempelajari pola yang kompleks [22, 23].

Seiring perkembangannya, *deep learning* telah melahirkan berbagai arsitektur yang dirancang untuk kebutuhan yang berbeda, seperti *Convolutional Neural Network* (CNN) untuk pengolahan citra, *Recurrent Neural Network* (RNN) untuk data sekuensial, dan *Generative Adversarial Network* (GAN) untuk pembangkitan data sintetis [21]. Di antara berbagai arsitektur tersebut, CNN menjadi pendekatan yang paling banyak digunakan dalam analisis citra karena mampu mengekstraksi fitur visual secara otomatis dan efektif, sehingga banyak diterapkan pada tugas *object detection*.

2.5 *Convolutional Neural Network*

Sebagai salah satu arsitektur yang banyak digunakan dalam *computer vision*, *Convolutional Neural Network* (CNN) dikembangkan untuk mengolah data yang tersusun dalam bentuk matriks, seperti citra digital. Berbeda dengan pendekatan

tradisional yang bergantung pada *feature engineering*, CNN mampu mempelajari fitur secara otomatis melalui proses pelatihan. Kemampuan tersebut memungkinkan model membangun representasi visual secara bertingkat, mulai dari pola sederhana hingga karakteristik objek yang lebih kompleks. Kemampuan tersebut menjadikan CNN sebagai salah satu metode yang banyak digunakan dalam berbagai tugas pengolahan citra, seperti klasifikasi citra, deteksi objek, dan segmentasi [21]. Selain itu, CNN memiliki mekanisme pembelajaran yang terstruktur, dengan proses ekstraksi fitur dilakukan secara bertahap seiring dengan bertambahnya kedalaman jaringan [22].



Gambar 2.5. *CNN Components*

Sumber: [22]

Pada Gambar 2.5, menjelaskan bahwa komponen *Convolutional Neural Network* (CNN) terdiri atas beberapa jenis lapisan (*layer*) yang masing-masing memiliki fungsi tertentu dalam proses pembelajaran representasi visual. Secara umum, CNN tersusun dari *convolution layer*, *activation layer*, *pooling layer*, serta *fully connected layer*. Kombinasi dari lapisan-lapisan tersebut memungkinkan model mempelajari fitur penting dari citra dan menghasilkan prediksi yang akurat.

1. *Convolution Layer*

   Pada CNN, *convolution layer* berfungsi untuk mengekstraksi informasi penting dari citra melalui penggunaan filter yang digeser pada area gambar.

Hasil proses tersebut berupa *feature map* yang memuat berbagai karakteristik visual, seperti tepi, tekstur, dan bentuk objek [21]. Efisiensi lapisan ini didukung oleh konsep *weight sharing*, yang memungkinkan sejumlah filter digunakan secara berulang tanpa menambah jumlah parameter secara signifikan [22].

2. *Activation Layer*

   Activation layer berperan dalam menambahkan sifat nonlinier pada jaringan sehingga model dapat mempelajari hubungan yang tidak dapat direpresentasikan oleh fungsi linear. ReLU (*Rectified Linear Unit*) merupakan salah satu fungsi aktivasi yang paling banyak digunakan karena efektif dalam mendukung proses pembelajaran serta meningkatkan kinerja model.

3. *Pooling Layer*

   Pooling layer digunakan untuk melakukan pengurangan dimensi spasial (*downsampling*) pada *feature map* yang dihasilkan oleh lapisan konvolusi. Tujuan utama proses ini adalah mengurangi kompleksitas komputasi sekaligus mempertahankan informasi penting yang telah dipelajari sebelumnya. Selain itu, *pooling* membantu meningkatkan ketahanan model terhadap perubahan kecil pada citra, seperti pergeseran posisi objek [21]. Metode yang umum diterapkan pada tahap ini adalah *max pooling* dan *average pooling*.

4. *Fully Connected Layer*

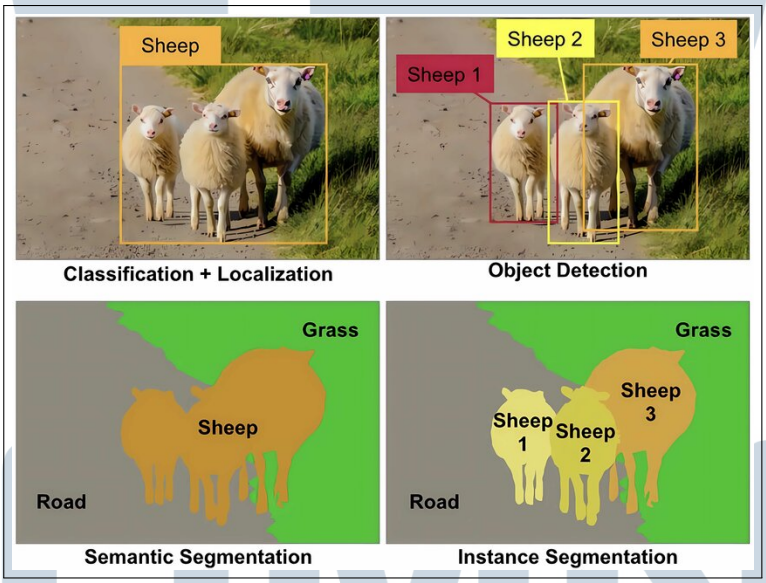
   Fully connected layer (FC) berperan sebagai lapisan klasifikasi pada tahap akhir jaringan. Pada lapisan ini, *feature map* yang telah diperoleh dari proses sebelumnya terlebih dahulu diratakan (*flatten*) menjadi vektor satu dimensi sebelum digunakan untuk menghasilkan prediksi kelas objek [21]. Pada beberapa arsitektur modern, termasuk YOLOv8, fungsi lapisan FC sebagian telah digantikan oleh *prediction head* yang secara langsung menghasilkan prediksi *bounding box* beserta nilai tingkat kepercayaan (*confidence score*) [22].

Kemampuan CNN dalam mengekstraksi fitur visual secara otomatis menjadi dasar pengembangan berbagai metode analisis citra, salah satunya *object detection*. Pada tugas ini, model tidak hanya mengklasifikasikan objek, tetapi juga menentukan lokasi setiap objek di dalam citra menggunakan *bounding box*. Representasi objek

melalui *bounding box* selanjutnya menjadi dasar dalam proses anotasi data yang digunakan pada pelatihan model *object detection*.

2.6 *Object Detection* dan Anotasi *Bounding Box*

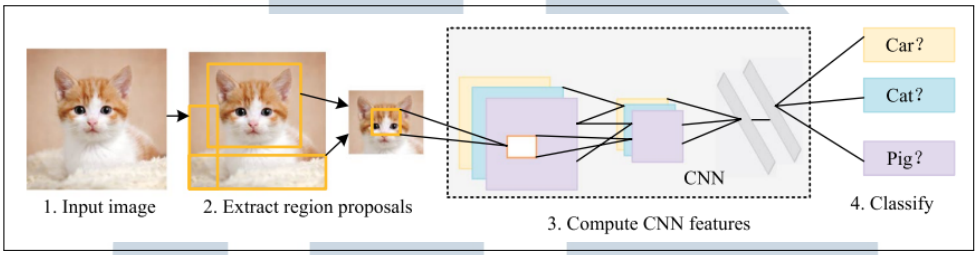
Dalam *computer vision*, *object detection* digunakan untuk mengenali keberadaan objek sekaligus menentukan letaknya pada suatu citra atau video. Proses ini menghasilkan informasi berupa kategori objek dan koordinat *bounding box* yang menunjukkan posisi objek dalam gambar [24]. Oleh karena itu, *object detection* menjadi salah satu teknologi yang banyak dimanfaatkan pada berbagai aplikasi yang memerlukan identifikasi objek secara otomatis.



Gambar 2.6. Ilustrasi Perbedaan Hasil *Classification*, *Object Detection*, dan *Segmentation*
Sumber: [18]

Tugas lain dalam *computer vision* meliputi *image classification* dan *segmentation*. *Image classification* hanya bertujuan menentukan kategori dari keseluruhan gambar tanpa memberikan informasi lokasi objek [25]. Sementara itu, *object detection* mampu melakukan klasifikasi objek sekaligus memprediksi koordinat *bounding box* pada setiap objek yang ditemukan. Adapun *segmentation* bekerja pada tingkat yang lebih rinci dengan mengidentifikasi setiap piksel yang termasuk ke dalam suatu objek, baik secara *semantic segmentation* maupun *instance segmentation* [26]. Perbedaan antara *image classification*, *object detection*, dan *segmentation* dapat dilihat pada Gambar 2.6.

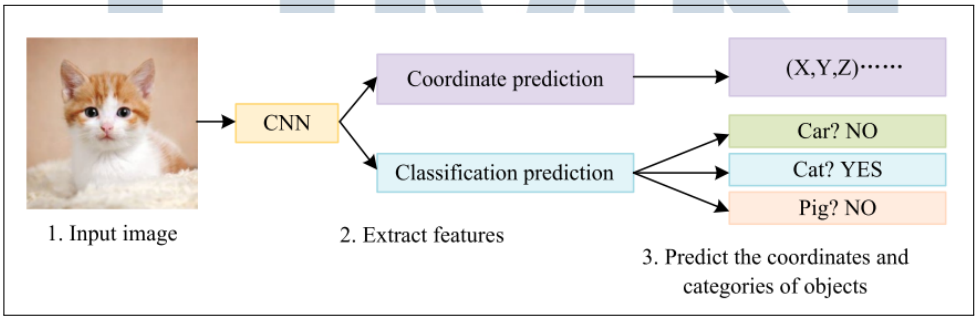
Perkembangan metode *object detection* mengalami perubahan yang signifikan, terutama setelah munculnya pendekatan berbasis *deep learning*. Secara umum, metode *object detection modern* dapat diklasifikasikan ke dalam dua kelompok utama, yaitu metode *two-stage* dan *one-stage* [27].



Gambar 2.7. Proses Dasar *Two-Stage Detector*

Sumber: [28]

Metode *two-stage* yang paling berpengaruh adalah keluarga *Region-based Convolutional Neural Network* (R-CNN), yang mencakup R-CNN, *Fast* R-CNN, dan *Faster* R-CNN. Pendekatan ini bekerja dengan menghasilkan sejumlah kandidat wilayah (*region proposal*) yang berpotensi mengandung objek, kemudian melakukan proses klasifikasi pada setiap wilayah tersebut secara terpisah. Meskipun mampu menghasilkan tingkat akurasi yang tinggi, metode *two-stage* cenderung memiliki waktu inferensi yang lebih lambat karena proses deteksi dilakukan melalui dua tahapan berurutan [24] Proses Dasar *Two-Stage Detector* dapat dilihat pada Gambar 2.7.



Gambar 2.8. Proses Dasar *One-Stage Detector*

Sumber: [28]

Untuk meningkatkan efisiensi proses deteksi, dikembangkan metode *one-stage* seperti SSD dan YOLO (*You Only Look Once*). Pendekatan ini melakukan lokalisasi dan klasifikasi objek secara bersamaan dalam satu kali proses inferensi sehingga mampu menghasilkan deteksi yang lebih cepat dibandingkan metode

two-stage [28]. Salah satu metode yang paling banyak digunakan adalah YOLO karena menawarkan keseimbangan yang baik antara kecepatan dan akurasi [25]. Proses Dasar *One-Stage Detector* dapat dilihat pada Gambar 2.8. Oleh sebab itu, YOLOv8 digunakan sebagai model object detection untuk proses identifikasi tablet paracetamol.

Pada proses *object detection*, anotasi *bounding box* memiliki peran penting dalam proses pelatihan model. Anotasi *bounding box* merupakan proses pelabelan posisi dan kelas objek pada setiap gambar dalam dataset. Setiap *bounding box* mendefinisikan wilayah persegi panjang yang membatasi objek di dalam gambar beserta label kelasnya. Akurasi anotasi menjadi faktor penting karena proses evaluasi model, seperti perhitungan *Intersection over Union* (IoU), bergantung langsung pada kualitas anotasi yang digunakan [29]. Penerapan *Bounding Box* dapat dilihat pada Gambar 2.9

Gambar 2.9. *Bounding Box*
Sumber: [29]

Format anotasi yang digunakan pada model YOLO berbeda dengan format lain seperti Pascal VOC atau COCO. Pada format YOLO, setiap gambar memiliki satu berkas teks (`.txt`) yang berisi informasi objek dengan bentuk sebagai berikut:

```
<class_id> <x_center> <y_center> <width> <height>
```

Keterangan dari masing-masing parameter ditunjukkan pada Tabel 2.3.

UNIVERSITAS MULTIMEDIA NUSANTARA

Tabel 2.3. Parameter Format Anotasi YOLO

Parameter	**Keterangan**
`class_id`	Indeks kelas objek dalam bentuk bilangan bulat dimulai dari 0.
`x_center`	Koordinat horizontal titik tengah *bounding box* yang dinormalisasi terhadap lebar gambar.
`y_center`	Koordinat vertikal titik tengah *bounding box* yang dinormalisasi terhadap tinggi gambar.
`width`	Menunjukkan nilai lebar *bounding box* yang telah dinormalisasi berdasarkan ukuran lebar citra.
`height`	Menunjukkan nilai tinggi *bounding box* yang telah dinormalisasi berdasarkan ukuran lebar citra.

Sebagai contoh, apabila suatu gambar berukuran 640×640 piksel memiliki objek dengan titik tengah pada koordinat $(320, 320)$ dan dimensi 100×80 piksel, maka anotasi dalam format YOLO ditulis sebagai berikut:

$$0 \quad 0.5000 \quad 0.5000 \quad 0.1563 \quad 0.1250$$

Penggunaan nilai yang dinormalisasi membuat format YOLO tidak bergantung pada resolusi gambar, sehingga anotasi tetap valid meskipun gambar mengalami proses *resizing* selama pelatihan model [30]. Setiap citra tablet paracetamol pada dataset dianotasi sesuai format YOLO untuk menyediakan informasi lokasi objek yang diperlukan oleh model YOLOv8.

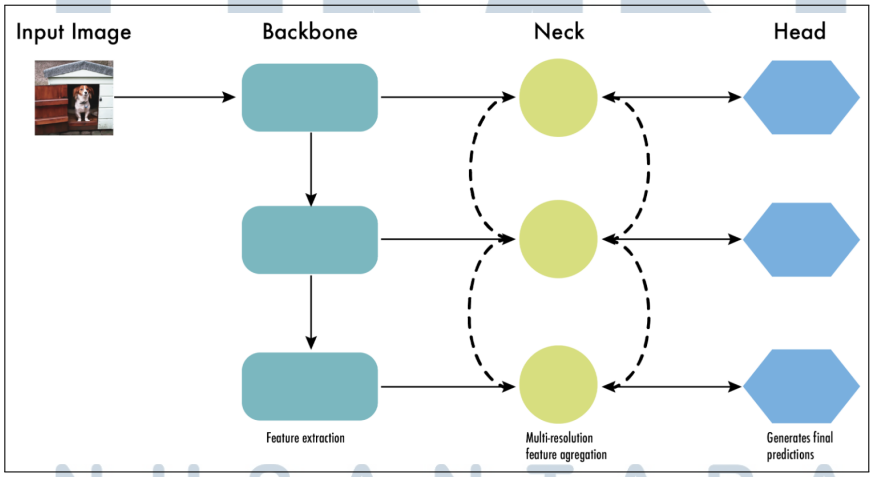
2.7 YOLO (*You Only Look Once*)

You Only Look Once (YOLO) merupakan sebuah model *object detection* berbasis *deep learning* yang diperkenalkan oleh Joseph Redmon dkk. pada tahun 2015. Kemunculan YOLO tidak lepas dari kebutuhan untuk mengatasi keterbatasan yang dimiliki oleh model *object detection* terdepan pada masa itu, yakni *Fast R-CNN*. Meskipun *Fast R-CNN* dikenal sebagai salah satu model paling akurat saat itu, model ini memiliki kelemahan mendasar dalam hal kecepatan pemrosesan. Untuk mengenali keberadaan objek sekaligus menentukan kelas objek pada citra yang digunakan sebagai input, *Fast R-CNN* membutuhkan waktu sekitar 2–3 detik per gambar, akibatnya, metode tersebut memiliki keterbatasan ketika digunakan pada

skenario yang menuntut proses deteksi secara *real time* [10].

YOLO hadir sebagai solusi atas permasalahan tersebut. Sesuai dengan namanya, konsep utama YOLO terletak pada pendekatannya yang menyelesaikan proses deteksi hanya dalam satu kali *forward pass* jaringan memproses gambar input secara keseluruhan sekaligus dan langsung menghasilkan prediksi *bounding box* serta klasifikasi kelas dalam satu langkah tunggal, tanpa memerlukan tahapan *region proposal* yang memakan waktu seperti pada keluarga R-CNN [31]. Hal ini menjadikan proses inferensi YOLO jauh lebih cepat dan efisien. Selain itu, arsitektur YOLO dirancang tanpa pipeline yang kompleks, sehingga model ini mampu memproses gambar pada kecepatan hingga 45 *frame per second* (fps). Dari sisi akurasi, YOLO mampu mencapai nilai *mean Average Precision* (mAP) dua kali lebih tinggi atau bahkan lebih dibandingkan model *real time object detection* lainnya yang tersedia pada saat itu, menjadikannya salah satu *state-of-the-art* untuk aplikasi yang membutuhkan pemrosesan cepat dan akurat secara bersamaan [10].

Keunggulan lain yang dimiliki YOLO adalah kemampuannya memproses gambar secara holistik. Karena seluruh gambar diproses sekaligus, bukan melalui pendekatan *sliding window* yang terfragmentasi. YOLO memiliki pemahaman kontekstual yang lebih baik terhadap objek dan lingkungan sekitarnya, sehingga cenderung menghasilkan lebih sedikit prediksi *false positive* [31]. YOLO juga terbukti memiliki kemampuan generalisasi yang baik ke berbagai domain baru, termasuk domain inspeksi objek di bidang farmasi.

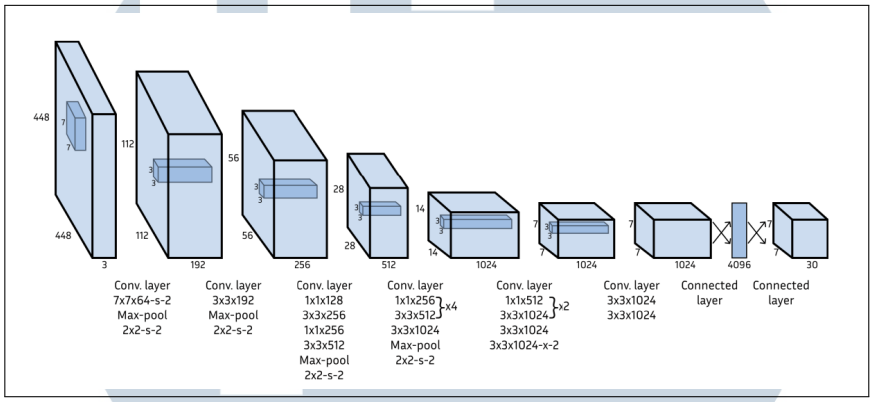


Gambar 2.10. Arsitektur *Modern Object Detector* untuk Estraksi Fitur dan Prediksi Objek

Sumber: [10]

Arsitektur YOLO secara umum terdiri atas tiga komponen utama,

yaitu *backbone*, *neck*, dan *head* [10]. Komponen *backbone* digunakan untuk mengekstraksi fitur visual dari citra, sedangkan *neck* berperan dalam menggabungkan informasi fitur pada berbagai skala. Hasil pemrosesan tersebut kemudian diteruskan ke *head*, yang bertugas menghasilkan keluaran berupa koordinat *bounding box*, skor kepercayaan, dan prediksi kelas objek.



Gambar 2.11. Arsitektur Awal YOLOv1

Sumber: [31]

Pada arsitektur awal YOLOv1 seperti yang diilustrasikan pada Gambar 2.11, jaringan dibangun dengan susunan lapisan sebagai berikut [10]:

- Input: Sebelum diproses oleh jaringan, citra terlebih dahulu diubah ukurannya menjadi 448×448 piksel.
- Lapisan Konvolusional (*Convolutional Layers*): YOLOv1 memanfaatkan 24 lapisan konvolusi untuk mengekstraksi informasi dari citra masukan. Konvolusi 1×1 digunakan untuk mereduksi jumlah *channel*, sedangkan konvolusi 3×3 berperan dalam mempelajari pola visual yang lebih kompleks. Hasil akhirnya berupa *feature map* dengan dimensi $7 \times 7 \times 1024$.
- Operasi *Max-Pooling*: YOLOv1 menggunakan beberapa lapisan *max-pooling* berukuran 2×2 dengan *stride* 2 untuk melakukan *downsampling*, sehingga ukuran representasi spasial berkurang secara bertahap dari 448×448 menjadi 7×7.
- Fungsi Aktivasi (*Activation Function*): Sebagian besar lapisan memanfaatkan fungsi aktivasi ReLU (*Rectified Linear Unit*), sedangkan lapisan output menggunakan aktivasi linear agar nilai prediksi dapat dihasilkan tanpa transformasi tambahan.

- Teknik Regularisasi: Digunakan pula teknik *batch normalization* untuk menstabilkan proses pelatihan dan mempercepat konvergensi, serta *dropout* untuk mencegah terjadinya *overfitting*.
- Lapisan *Fully Connected*: *Feature map* hasil ekstraksi fitur diratakan (*flatten*) dan diteruskan ke lapisan *fully connected* berukuran 4096 neuron sebelum menghasilkan output akhir.
- Output: Keluaran YOLOv1 berupa tensor berukuran $7 \times 7 \times 30$. Pada setiap *grid cell*, model memprediksi dua *bounding box* yang terdiri atas koordinat lokasi objek dan *confidence score*, serta probabilitas kemunculan 20 kelas objek.

Kecepatan YOLO diperoleh dari pendekatannya yang tidak memerlukan proses pencarian *Region of Interest* (ROI) secara terpisah. Sebagai gantinya, citra dibagi menjadi beberapa *grid cell* yang secara langsung melakukan prediksi lokasi dan kelas objek. Pada YOLOv1, citra berukuran 448×448 piksel dibagi menjadi grid 7×7, dengan setiap sel bertanggung jawab menghasilkan prediksi *bounding box* dan kategori objek pada area yang diamatinya [31].

Setiap *bounding box* pada YOLO direpresentasikan oleh koordinat pusat (x, y), ukuran kotak (w, h), dan *confidence score*. Nilai *confidence score* digunakan sebagai indikator tingkat keyakinan model terhadap keberadaan objek pada area yang diprediksi. Semakin tinggi nilainya, semakin besar keyakinan model bahwa objek berhasil terdeteksi dengan lokasi yang sesuai. Selain memprediksi *bounding box*, setiap *grid cell* juga menghasilkan probabilitas kelas yang digunakan untuk menentukan kategori objek yang terdeteksi [10].

Karena setiap *cell* grid menghasilkan banyak kandidat *bounding box* secara bersamaan, output awal YOLO berpotensi mengandung sangat banyak kotak prediksi yang saling tumpang tindih untuk objek yang sama. Untuk menyaring dan menentukan prediksi final, digunakan mekanisme *Non-Maximum Suppression* (NMS).

UNIVERSITAS MULTIMEDIA NUSANTARA

Gambar 2.12. Non-Maximum Suppression

Sumber: [10]

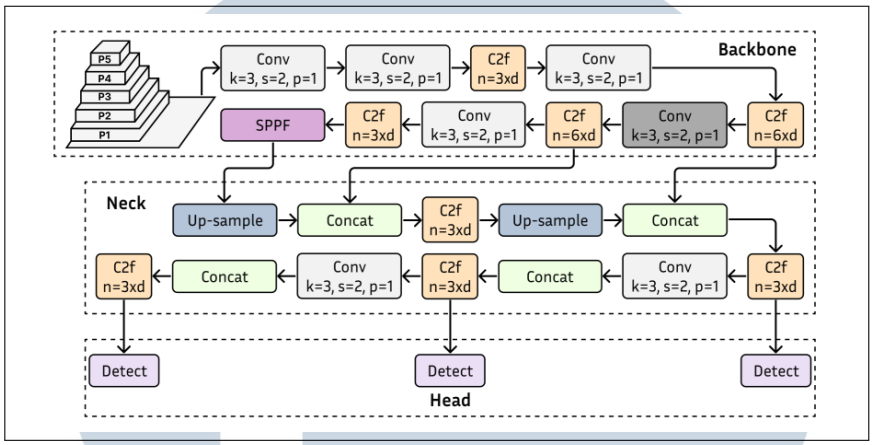
Proses NMS bekerja dalam dua tahap seleksi. Pertama, seluruh *bounding box* yang memiliki *confidence score* di bawah nilai ambang batas (*confidence threshold*) yang telah ditentukan akan langsung diabaikan dan dieliminasi, terlepas dari kelas objek apa pun yang diprediksinya. Kedua, dari *bounding box* yang tersisa dan masih saling tumpang tindih, dilakukan seleksi lebih lanjut menggunakan nilai IoU. Apabila dua *bounding box* memiliki nilai IoU yang melebihi *threshold IoU* yang ditentukan, artinya keduanya mendeteksi objek yang sama maka *bounding box* dengan nilai *confidence* yang lebih rendah akan dieliminasi. Proses ini dilakukan secara per kelas dan diulang secara iteratif hingga setiap objek hanya diwakili oleh satu *bounding box* terbaik, sehingga output deteksi akhir menjadi bersih dan tidak redundan [31]. Ilustrasi proses NMS dapat dilihat pada Gambar 2.12.

Sejak pertama kali diperkenalkan, YOLO telah mengalami evolusi yang berkelanjutan melalui berbagai versi. Mulai dari YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, hingga versi-versi terkini. dengan setiap iterasi membawa peningkatan signifikan pada akurasi, kecepatan, maupun efisiensi arsitektur [10] [31].

2.8 YOLOv8

YOLOv8 (*You Only Look Once version 8*) merupakan model *object detection* yang dikembangkan oleh *Ultralytics* dan dirilis pada Januari 2023. Berbeda dengan YOLO versi sebelumnya, YOLOv8 menghadirkan perancangan ulang arsitektur (*architectural redesign*) melalui penggunaan modul C2f pada *backbone*, pendekatan *anchor-free detection*, serta *decoupled detection head*. Perubahan tersebut bertujuan meningkatkan akurasi deteksi, efisiensi komputasi,

dan kemampuan generalisasi model pada berbagai dataset [10]. Selain *object detection*, YOLOv8 juga mendukung *instance segmentation*, *pose estimation*, dan *image classification* dalam satu kerangka kerja [32].



Gambar 2.13. Arsitektur YOLOv8

Sumber: [31]

Secara umum, YOLOv8 memproses citra masukan dalam satu tahap (*single-stage detector*). Sebagai contoh, citra berukuran 640×640 piksel terlebih dahulu diproses oleh *backbone*. Melalui beberapa lapisan konvolusi dengan *stride* 2, ukuran *feature map* diperkecil secara bertahap menjadi 320×320, 160×160, 80×80, 40×40, hingga 20×20. Meskipun resolusi spasial semakin kecil, jumlah kanal fitur bertambah sehingga model mampu mempelajari representasi visual yang semakin kompleks. Selanjutnya, *feature map* dari berbagai skala digabungkan pada bagian *neck* sebelum diteruskan ke *detection head* untuk menghasilkan prediksi kelas objek, koordinat *bounding box*, dan nilai *confidence score*.

Arsitektur YOLOv8 terdiri atas tiga komponen utama sebagaimana ditunjukkan pada Gambar 2.13, yaitu:

1. *Backbone*

   Backbone bertugas mengekstraksi fitur visual dari citra masukan. YOLOv8 menggunakan modul C2f (*Cross Stage Partial Bottleneck with Two Convolutions*) sebagai pengembangan dari modul C3 pada YOLOv5. Modul ini menghasilkan aliran gradien yang lebih baik sehingga meningkatkan kualitas representasi fitur tanpa menambah kompleksitas komputasi secara signifikan. Pada bagian akhir *backbone*, digunakan modul *Spatial Pyramid Pooling Fast* (SPPF) untuk memperoleh fitur multi-skala secara efisien [10, 32].

2. *Neck*

   Bagian *neck* menggunakan arsitektur PAN-FPN (*Path Aggregation Network Feature Pyramid Network*) untuk menggabungkan informasi fitur dari berbagai tingkat resolusi. Proses ini memungkinkan model mendeteksi objek berukuran kecil, sedang, maupun besar dengan lebih akurat [31].

3. *Detection Head*

   YOLOv8 menggunakan pendekatan *anchor-free* sehingga koordinat *bounding box* diprediksi secara langsung tanpa memerlukan *anchor box*. Selain itu, YOLOv8 menerapkan *decoupled head*, yaitu pemisahan proses klasifikasi dan regresi *bounding box* ke dalam cabang yang berbeda. Pendekatan tersebut meningkatkan efektivitas pembelajaran sehingga menghasilkan performa deteksi yang lebih baik dibandingkan pendekatan *coupled head* pada versi sebelumnya [10].

YOLOv8 tersedia dalam lima varian model, yaitu YOLOv8n (*nano*), YOLOv8s (*small*), YOLOv8m (*medium*), YOLOv8l (*large*), dan YOLOv8x (*extra large*). Kelima model memiliki arsitektur yang sama, namun berbeda pada jumlah parameter, kompleksitas komputasi, kebutuhan memori, serta tingkat akurasi yang dihasilkan. Model yang lebih kecil memiliki kecepatan inferensi lebih tinggi, sedangkan model yang lebih besar umumnya memberikan akurasi yang lebih baik dengan kebutuhan komputasi yang lebih tinggi [10]. Perbandingan karakteristik masing-masing model ditunjukkan pada Tabel 2.4.

Tabel 2.4. Karakteristik Model YOLOv8

Model	**Karakteristik**	**Penggunaan**
YOLOv8n	Paling ringan dan tercepat	Mobile/Edge
YOLOv8s	Ringan, akurasi lebih baik	Embedded
YOLOv8m	Seimbang antara akurasi dan komputasi	aplikasi umum
YOLOv8l	Akurasi tinggi, komputasi besar	GPU tinggi
YOLOv8x	Akurasi tertinggi, model terbesar	Server/HPC

YOLOv8m dipilih karena memberikan keseimbangan antara akurasi dan efisiensi komputasi. Dibandingkan YOLOv8n dan YOLOv8s, model ini memiliki

kemampuan representasi fitur yang lebih baik untuk membedakan karakteristik visual tablet yang serupa. Sementara itu, YOLOv8l dan YOLOv8x membutuhkan sumber daya komputasi yang lebih besar dengan peningkatan akurasi yang relatif tidak signifikan untuk dataset yang digunakan. Selain itu, YOLOv8m telah menunjukkan performa yang baik pada berbagai penelitian *object detection* dengan jumlah kelas terbatas [33, 31].

Berdasarkan arsitektur tersebut, YOLOv8 menghasilkan prediksi berupa kelas objek, lokasi *bounding box*, dan nilai *confidence score*. Kinerja model kemudian diukur menggunakan berbagai metrik evaluasi, seperti *Precision*, *Recall*, *F1-Score*, *Intersection over Union* (IoU), *Average Precision* (AP), dan *mean Average Precision* (mAP). Metrik-metrik tersebut digunakan untuk menilai kemampuan model dalam mendeteksi dan mengklasifikasikan objek secara akurat.

2.9 Metrik Evaluasi Model

Pengukuran performa model dilakukan melalui tahap evaluasi menggunakan sejumlah metrik yang umum digunakan pada penelitian *object detection*. Evaluasi model dilakukan untuk mengukur kemampuan model dalam mengenali dan mendeteksi objek pada data uji yang tidak terlibat selama proses pelatihan. Kinerja model dievaluasi menggunakan beberapa metrik, yaitu *Confusion Matrix*, akurasi, *Precision*, *Recall*, *F1-Score*, dan mAP [34]. Nilai yang diperoleh dari setiap metrik digunakan untuk menilai tingkat performa serta efektivitas model yang dikembangkan.

2.9.1 *Confusion Matrix*

Dalam proses evaluasi model, *Confusion Matrix* dimanfaatkan untuk menganalisis hasil klasifikasi dengan membandingkan keluaran model terhadap label referensi. Matriks ini memberikan gambaran mengenai jumlah prediksi yang sesuai maupun yang tidak sesuai dengan label sebenarnya pada setiap kelas [34].

Confusion Matrix terdiri dari empat komponen utama, yaitu *True Positive* (TP), *False Positive* (FP), *True Negative* (TN), dan *False Negative* (FN), yang selanjutnya digunakan sebagai dasar dalam proses evaluasi model.

Tabel 2.5. Confusion Matrix

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

Berbagai metrik evaluasi, termasuk *Accuracy*, *Precision*, *Recall*, dan *F1-Score*, dihitung berdasarkan kombinasi nilai TP, TN, FP, dan FN yang diperoleh dari *Confusion Matrix* [34].

2.9.2 *Accuracy*

Dalam evaluasi model, *accuracy* digunakan untuk mengetahui tingkat ketepatan prediksi secara keseluruhan. Metrik ini menunjukkan perbandingan antara jumlah prediksi yang benar dengan total data yang dievaluasi. Semakin besar nilai *accuracy*, semakin baik kemampuan model dalam menghasilkan prediksi yang sesuai dengan kondisi sebenarnya [34].

Perhitungan nilai *accuracy* dilakukan menggunakan Persamaan 2.2.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.2}$$

2.9.3 *Precision*

Precision digunakan untuk menilai tingkat keakuratan prediksi positif yang dihasilkan model. Metrik ini menggambarkan seberapa banyak prediksi positif yang sesuai dengan kondisi sebenarnya dibandingkan seluruh prediksi positif yang dibuat oleh model. Nilai *precision* yang tinggi menunjukkan bahwa jumlah kesalahan prediksi positif relatif rendah [34].

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

Pada Persamaan 2.3, perhitungan *precision* didasarkan pada nilai TP dan FP. Nilai yang mendekati 1 menunjukkan bahwa sebagian besar prediksi positif yang dihasilkan model sesuai dengan kondisi sebenarnya.

2.9.4 *Recall*

Recall digunakan untuk mengukur tingkat keberhasilan model dalam mendeteksi objek positif pada data. Nilai ini menggambarkan proporsi objek yang berhasil ditemukan dibandingkan dengan seluruh objek positif yang tersedia [34].

$$Recall = \frac{TP}{TP + FN} \tag{2.4}$$

Nilai *recall* yang tinggi menunjukkan bahwa model mampu mendeteksi sebagian besar objek yang ada pada data pengujian, sedangkan nilai yang rendah mengindikasikan masih banyak objek yang terlewat.

2.9.5 *F1-Score*

F1-Score merupakan metrik evaluasi yang digunakan untuk mengukur keseimbangan antara nilai *precision-recall*. Metrik ini memberikan gambaran performa model secara keseluruhan dengan mempertimbangkan kedua aspek tersebut secara bersamaan [34].

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.5}$$

Nilai *F1-Score* berada pada rentang 0 hingga 1. Nilai yang semakin mendekati 1 menunjukkan bahwa model memiliki keseimbangan yang baik antara ketepatan prediksi dan kemampuan mendeteksi objek.

2.9.6 *Intersection over Union* (IoU)

Intersection over Union (IoU), Merupakan rasio antara luas area irisan (*intersection*) dan luas area gabungan (*union*) dari *bounding box* hasil prediksi dengan *ground truth*, untuk menentukan apakah suatu prediksi *bounding box* termasuk benar (*True Positive*) atau salah.[31].

$$IoU = \frac{Area(B_{pred} \cap B_{gt})}{Area(B_{pred} \cup B_{gt})} \tag{2.6}$$

Pada Persamaan 2.6, B_{pred} merupakan *bounding box* hasil prediksi model,

sedangkan B_{gt} merupakan *ground truth*. Nilai IoU berada pada rentang 0 hingga 1, di mana nilai yang semakin mendekati 1 menunjukkan tingkat kesesuaian yang semakin tinggi antara hasil prediksi dan *ground truth*. Pada evaluasi YOLO, suatu prediksi umumnya dikategorikan sebagai *True Positive* apabila nilai IoU memenuhi atau melebihi ambang batas (*threshold*) yang telah ditentukan.

2.9.7 *Average Precision*

Average Precision (AP) merupakan metrik yang digunakan untuk mengukur performa deteksi pada setiap kelas objek. Nilai AP diperoleh dari luas area di bawah kurva *Precision-Recall*, sehingga mencerminkan keseimbangan antara ketepatan prediksi (*precision*) dan kemampuan model dalam menemukan seluruh objek (*recall*) pada suatu kelas [31].

$$AP = \int_0^1 P(R), dR \tag{2.7}$$

Pada Persamaan 2.7, $P(R)$ menyatakan nilai *precision* pada setiap nilai *recall*. Dalam implementasi praktis, termasuk pada YOLOv8, nilai AP dihitung secara numerik dari kurva *Precision-Recall* yang dihasilkan berdasarkan hasil deteksi pada suatu kelas objek.

2.9.8 *Mean Average Precision*

Mean Average Precision (mAP) merupakan metrik evaluasi yang sering digunakan untuk mengukur performa model *object detection*. Nilai mAP diperoleh dari rata-rata nilai *Average Precision*(AP) pada seluruh kelas objek yang digunakan dalam pengujian [31].

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{2.8}$$

Pada Persamaan 2.8, N menyatakan jumlah kelas objek, sedangkan AP_i merupakan nilai *Average Precision* pada kelas ke-i.

Dalam evaluasi YOLOv8, dua metrik mAP yang umum digunakan adalah mAP@50 dan mAP@50-95. Nilai mAP@50 dihitung menggunakan ambang batas IoU sebesar 0,50, sehingga suatu prediksi dianggap benar apabila memiliki nilai

IoU minimal 0,50. Sementara itu, mAP@50-95 menghitung rata-rata AP pada sepuluh nilai ambang IoU, yaitu 0,50 hingga 0,95 dengan interval 0,05. Oleh karena itu, mAP@50-95 memberikan evaluasi yang lebih ketat karena menilai tidak hanya keberhasilan mendeteksi objek, tetapi juga ketepatan posisi *bounding box* hasil prediksi terhadap *ground truth*. Semakin tinggi nilai mAP@50 maupun mAP@50-95, semakin baik performa model dalam melakukan deteksi objek secara akurat.

2.10 *Cross Validation & Stratified K-Fold Cross Validation*

Cross Validation merupakan teknik validasi yang digunakan untuk mengevaluasi kemampuan generalisasi model *machine learning* terhadap data yang belum pernah digunakan pada proses pelatihan. Metode ini membagi dataset menjadi beberapa bagian (*fold*), kemudian secara bergantian menggunakan satu *fold* sebagai data pengujian dan sisanya sebagai data pelatihan. Nilai performa model diperoleh dari rata-rata hasil evaluasi seluruh iterasi sehingga memberikan estimasi yang lebih representatif [35]

2.10.1 Stratified K-Fold Cross Validation

Stratified K-Fold Cross Validation merupakan pengembangan dari *K-Fold Cross Validation* yang mempertahankan proporsi setiap kelas pada setiap *fold* sesuai dengan distribusi data asli. Pendekatan ini sesuai digunakan pada permasalahan klasifikasi karena menghasilkan pembagian data yang lebih seimbang, sehingga evaluasi model menjadi lebih stabil dan akurat [36, 35].

Stratified K-Fold Cross Validation digunakan untuk membagi data pelatihan dan pengujian dengan tetap mempertahankan proporsi setiap kelas pada setiap *fold*, sehingga hasil evaluasi model dapat menggambarkan kemampuan generalisasi secara lebih baik.

2.11 SUS (*System Usability Scale*)

System Usability Scale (SUS) merupakan instrumen evaluasi *usability* yang dikembangkan oleh Brooke pada tahun 1986 untuk mengukur kemudahan penggunaan suatu sistem berdasarkan persepsi pengguna. SUS banyak digunakan dalam evaluasi berbagai produk digital karena sederhana, mudah dipahami, serta mampu memberikan gambaran umum mengenai tingkat *usability* hanya melalui sepuluh butir pertanyaan [37].

SUS dipilih karena berfokus pada evaluasi *usability*, seperti kemudahan dipelajari (*learnability*), kemudahan digunakan (*ease of use*), efisiensi, dan tingkat kepercayaan pengguna saat berinteraksi dengan sistem. Berbeda dengan metode seperti *End User Computing Satisfaction* (EUCS) yang menilai kepuasan pengguna terhadap sistem informasi, SUS lebih sesuai digunakan untuk mengevaluasi kemudahan penggunaan aplikasi identifikasi tablet paracetamol berbasis web yang dikembangkan. Selain itu, jumlah pertanyaan yang relatif sedikit membuat proses evaluasi menjadi lebih cepat tanpa mengurangi kemampuan instrumen dalam memberikan gambaran kualitas *usability* [38, 39].

SUS dikenal sebagai instrumen evaluasi *usability* yang dapat diterapkan pada berbagai ukuran sampel, termasuk jumlah responden yang relatif kecil. Meskipun demikian, ukuran sampel tetap memengaruhi tingkat kepercayaan dan stabilitas skor yang diperoleh. Semakin banyak responden yang dilibatkan, semakin representatif hasil evaluasi terhadap populasi pengguna dan semakin kecil ketidakpastian hasil survei. Oleh karena itu, jumlah responden dalam evaluasi SUS perlu disesuaikan dengan tujuan penelitian, karakteristik pengguna, ketersediaan responden, serta tingkat representativitas hasil yang ingin dicapai [40].

Instrumen SUS terdiri atas 10 butir pertanyaan yang dijawab menggunakan skala Likert lima tingkat, mulai dari nilai 1 (*sangat tidak setuju*) hingga 5 (*sangat setuju*). Untuk mengurangi kecenderungan responden memberikan jawaban yang seragam (*response bias*), susunan pertanyaan disusun secara bergantian antara pernyataan positif dan negatif. Pertanyaan bernomor ganjil (1, 3, 5, 7, dan 9) merupakan pernyataan positif, sedangkan pertanyaan bernomor genap (2, 4, 6, 8, dan 10) merupakan pernyataan negatif [38, 39].

SUS digunakan untuk memperoleh tanggapan pengguna setelah mencoba seluruh fitur utama aplikasi, sehingga skor yang dihasilkan dapat menggambarkan tingkat kemudahan penggunaan sistem secara langsung.

2.11.1 Cara Perhitungan Skor SUS

Perhitungan skor SUS dilakukan melalui tahapan sebagai berikut.

1. Setiap pertanyaan dinilai menggunakan skala Likert dengan rentang skor 1–5.
2. Untuk pernyataan positif (nomor 1, 3, 5, 7, dan 9), nilai kontribusi dihitung

menggunakan Persamaan 2.9.

$$Nilai\ Kontribusi = (Jawaban - 1) \tag{2.9}$$

3. Untuk pernyataan negatif (nomor 2, 4, 6, 8, dan 10), nilai kontribusi dihitung menggunakan Persamaan 2.10.

$$Nilai\ Kontribusi = (5 - Jawaban) \tag{2.10}$$

4. Seluruh nilai kontribusi dijumlahkan untuk memperoleh skor mentah.
5. Skor SUS dihitung dengan mengalikan skor mentah dengan faktor 2,5 sesuai Persamaan 2.11.

$$Skor\ SUS = (Total\ Nilai\ Kontribusi) \times 2.5 \tag{2.11}$$

Nilai SUS berada pada rentang 0–100. Secara umum, skor di atas 68 menunjukkan tingkat *usability* yang baik, sedangkan skor di atas 80 menunjukkan tingkat *usability* yang sangat baik (*excellent usability*) [38, 39, 41].

2.12 Skala Likert

Pengumpulan penilaian dari responden dilakukan berdasarkan Skala Likert. Melalui Skala Likert, tanggapan responden terhadap suatu pernyataan dapat direpresentasikan dalam bentuk skor numerik, sehingga hasil yang diperoleh dapat diolah dan dianalisis secara statistik [37].

Skala Likert yang diterapkan ini menggunakan lima tingkat penilaian, mulai dari Sangat Tidak Setuju (STS) hingga Sangat Setuju (SS). Setiap tingkat penilaian dikonversi ke dalam skor numerik yang mencerminkan tingkat persetujuan responden terhadap pernyataan yang diberikan. Distribusi skor untuk setiap kategori ditampilkan pada Tabel 2.6.

Tabel 2.6. Skala Penilaian Likert

Pilihan Jawaban	Skor
Sangat Tidak Setuju (STS)	1
Tidak Setuju (TS)	2
Netral (N)	3
Setuju (S)	4
Sangat Setuju (SS)	5

Pemberian skor dilakukan berdasarkan tingkat persetujuan responden terhadap setiap pernyataan. Nilai yang diperoleh dari setiap respons responden menjadi dasar dalam melakukan penilaian terhadap kualitas penggunaan sistem [37].

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA