BAB 2
LANDASAN TEORI

2.1 Tinjauan Teori

2.1.1 Deteksi Jatuh

Dalam penelitian deteksi jatuh sebelumnya, terdapat beberapa pendekatan yang telah diusulkan, seperti penggunaan YOLOv3 [7], pendekatan berbasis lightweight CGNS-YOLO [8], dan metode berbasis estimasi pose dan YOLOv5 [12]. YOLOv3 telah terbukti efektif dalam mendeteksi kejadian jatuh pada lansia dengan tingkat akurasi yang memuaskan [7]. Namun, pendekatan ini masih memiliki kendala dalam hal kompleksitas dan kebutuhan komputasi yang tinggi. Di sisi lain, CGNS-YOLO menawarkan pendekatan yang lebih ringan dengan mengurangi ukuran model dan meningkatkan efisiensi deteksi [8]. Metode berbasis estimasi pose dan YOLOv5 juga menunjukkan kemampuan yang baik dalam mendeteksi perilaku jatuh dalam video pengawasan [12]. YOLOv8 diharapkan mampu mengatasi beberapa kendala yang dimiliki oleh pendekatan sebelumnya, seperti kompleksitas dan kebutuhan komputasi yang tinggi dari YOLOv3, serta memberikan keseimbangan antara akurasi deteksi dan efisiensi komputasi seperti yang ditawarkan oleh CGNS-YOLO dan metode berbasis estimasi pose dan YOLOv5. Dengan demikian, YOLOv8 diharapkan dapat menjadi solusi yang lebih efektif dan efisien dalam mendeteksi kejadian jatuh, memberikan kontribusi yang signifikan dalam upaya pencegahan cedera dan respons cepat terhadap kejadian jatuh.

2.1.2 Machine Learning

Machine learning adalah salah satu pembelajaran ilmiah tentang algoritma dan model statistik yang digunakan oleh sistem komputer untuk melakukan tugas spesifik tanpa diprogram secara spesifik [13]. *Deep learning* adalah salah satu konsep dalam *machine learning* yang berbasis pada *artificial neural networks* dan sering melampaui kemampuan analisis data tradisional lainnya, memungkinkan sistem untuk melakukan otomatisasi pembangunan model analisis dan menyelesaikan tugas-tugas kompleks dengan mempelajari data pelatihan yang spesifik [14]. *Artificial neural networks* (ANNs) adalah salah satu model komputasi

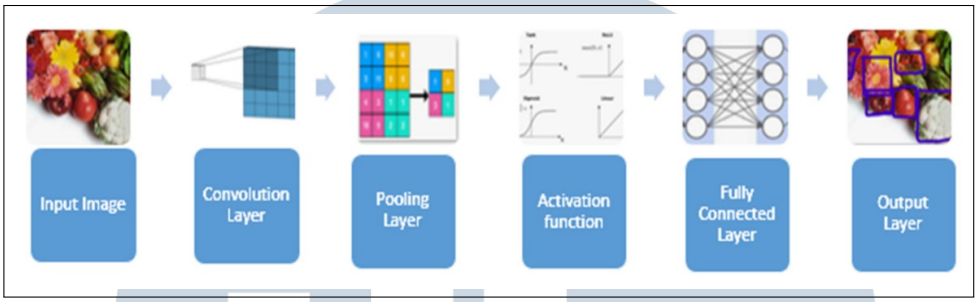
yang terinspirasi dari jaringan saraf otak, menyediakan alat yang sangat kuat untuk analisis data dan memungkinkan para ilmuwan saraf untuk memodelkan perilaku kompleks dan pola aktivitas saraf [15]. *Deep learning* menambahkan lapisan-lapisan tersembunyi di antara lapisan *input* dan *output* pada *Convolutional Neural Networks* yang bertujuan untuk menunjukkan model hubungan yang kompleks dan nonlinier [16].

Meskipun *machine learning* dan *deep learning* sering digunakan secara bergantian, keduanya memiliki perbedaan penting. *Machine learning* adalah teknik di mana sistem komputer belajar dari data untuk membuat keputusan atau prediksi tanpa diprogram secara eksplisit. Algoritma dalam *machine learning* meningkatkan kinerjanya seiring waktu saat menerima lebih banyak data. Di sisi lain, *deep learning* adalah bagian dari *machine learning* yang menggunakan jaringan saraf buatan yang meniru cara kerja otak manusia. *Deep learning* sangat efektif untuk tugas-tugas yang kompleks, seperti pengenalan gambar atau suara, dan seringkali memberikan hasil yang lebih baik dibandingkan teknik *machine learning* tradisional. Secara sederhana, *machine learning* adalah metode umum untuk membuat komputer belajar dari data, sementara *deep learning* adalah teknik canggih dalam *machine learning* yang menggunakan jaringan saraf untuk menangani masalah yang lebih kompleks [14].

2.1.3 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) adalah salah satu tipe *neural network* yang menggunakan operasi *convolution* atau lilitan sebagai layer fundamentalnya, bukan hanya mengandalkan pada lapisan-lapisan yang terhubung saja. CNNs unggul dalam tugas yang melibatkan data masukan dengan struktur grid, seperti deret waktu atau gambar, dan telah memainkan peran penting dalam kemajuan dalam pembelajaran mendalam, terutama dalam aplikasi visi komputer [17]. CNNs dikenal karena kemampuannya untuk secara otomatis belajar dan meningkat dari data tanpa memerlukan teknik rekayasa fitur manual. CNN dapat mengelola data tidak terstruktur, seperti teks dan gambar, dan secara otomatis mengekstrak fitur, menghilangkan kebutuhan untuk pra-pemrosesan data. CNN adalah variasi dari *Multilayer Perceptron* yang terinspirasi oleh jaringan saraf manusia berdasarkan penemuan Hubel dan Wiesel yang melakukan penelitian korteks visual pada indra visual kucing. Studi ini sangat terinspirasi oleh cara kerjanya karena dalam hasil penelitian, korteks visual pada hewan sangat kuat

dalam sistem pemrosesan visual yang pernah ada [18]. Komponen CNN dapat dilihat pada Gambar 2.1.



Gambar 2.1. Komponen CNN
[19]

A Convolutional Layer

Layer konvolusi adalah komponen penting dari CNN, yang utamanya menggunakan kernel yang dapat dipelajari untuk melakukan konvolusi di sepanjang dimensi spasial input, menghasilkan peta aktivasi 2D yang memvisualisasikan kehadiran fitur. Setiap kernel menghitung produk skalar untuk nilainya, belajar untuk mengaktifkan saat mendeteksi fitur tertentu pada posisi spasial tertentu. Aktivasi ini membentuk peta aktivasi yang sesuai, ditumpuk untuk membuat volume output layer, dengan neuron terhubung ke wilayah input yang kecil, yang didefinisikan sebagai ukuran bidang reseptif. Parameter seperti kedalaman, langkah, dan zero-padding mengoptimalkan kompleksitas output, dengan zero-padding mengendalikan dimensi volume output. Teknik-teknik ini mengubah dimensi spasial output layer, dihitung menggunakan rumus tertentu. Berbagi parameter mengurangi parameter secara signifikan dengan membatasi setiap peta aktivasi ke bobot dan bias yang sama, menyederhanakan backpropagasi dengan memperbarui satu set bobot secara keseluruhan [20].

B Pooling Layer

Layer pooling bertujuan untuk secara bertahap mengurangi dimensi representasi, serta mengurangi jumlah parameter dan kompleksitas komputasi model lebih lanjut. Layer ini beroperasi pada setiap peta aktivasi dalam input, dan menyesuaikan dimensinya menggunakan fungsi ”MAX”. Pada kebanyakan CNN, ini berbentuk layer max-pooling dengan kernel berdimensi 2 × 2 yang

diterapkan dengan langkah 2 sepanjang dimensi spasial input. Ini mengurangi skala peta aktivasi menjadi 25% dari ukuran aslinya - sambil mempertahankan volume kedalaman ke ukuran standarnya. Karena sifat destruktif layer pooling, umumnya hanya ada dua metode yang diamati untuk max-pooling. Biasanya, langkah dan filter layer pooling keduanya diatur menjadi 2×2, yang akan memungkinkan layer untuk meluas melalui seluruh dimensi spasial input. Selain itu, pooling tumpang tindih juga dapat digunakan, di mana langkah diatur menjadi 2 dengan ukuran kernel diatur menjadi 3. Karena sifat destruktif pooling, memiliki ukuran kernel di atas 3 biasanya akan sangat mengurangi kinerja model.

Penting untuk memahami bahwa di luar max-pooling, arsitektur CNN mungkin juga mengandung pooling umum. Layer pooling umum terdiri dari neuron pooling yang mampu melakukan berbagai operasi umum termasuk normalisasi L1/L2, dan pooling rata-rata. Namun, tutorial ini akan lebih fokus pada penggunaan max-pooling [20].

C Activation Function

Dalam Jaringan Saraf Konvolusi (CNN), lapisan aktivasi memainkan peran penting dengan menerapkan fungsi aktivasi pada keluaran lapisan konvolusi, memperkenalkan non-linearitas ke dalam jaringan dan memungkinkannya untuk mempelajari pola-pola kompleks dalam data. Fungsi aktivasi dapat memetakan nilai keluaran ke rentang tertentu, seperti -1 hingga 1 atau 0 hingga 1, yang penting untuk tugas-tugas seperti klasifikasi. Ada dua jenis utama fungsi aktivasi: linear dan non-linear. Fungsi aktivasi linear, seperti F(x)=cx, menghasilkan keluaran yang proporsional dengan masukan, sementara fungsi aktivasi non-linear, seperti ReLU (Rectified Linear Unit), lebih umum digunakan dalam jaringan modern karena memungkinkan pemodelan data yang rumit dan berdimensi tinggi. ReLU, khususnya, populer dalam CNN karena kesederhanaannya dan efektivitasnya, mengeluarkan masukan secara langsung jika positif dan nol jika tidak, sehingga membantu dalam ekstraksi fitur yang efisien dan pembelajaran dari data [19].

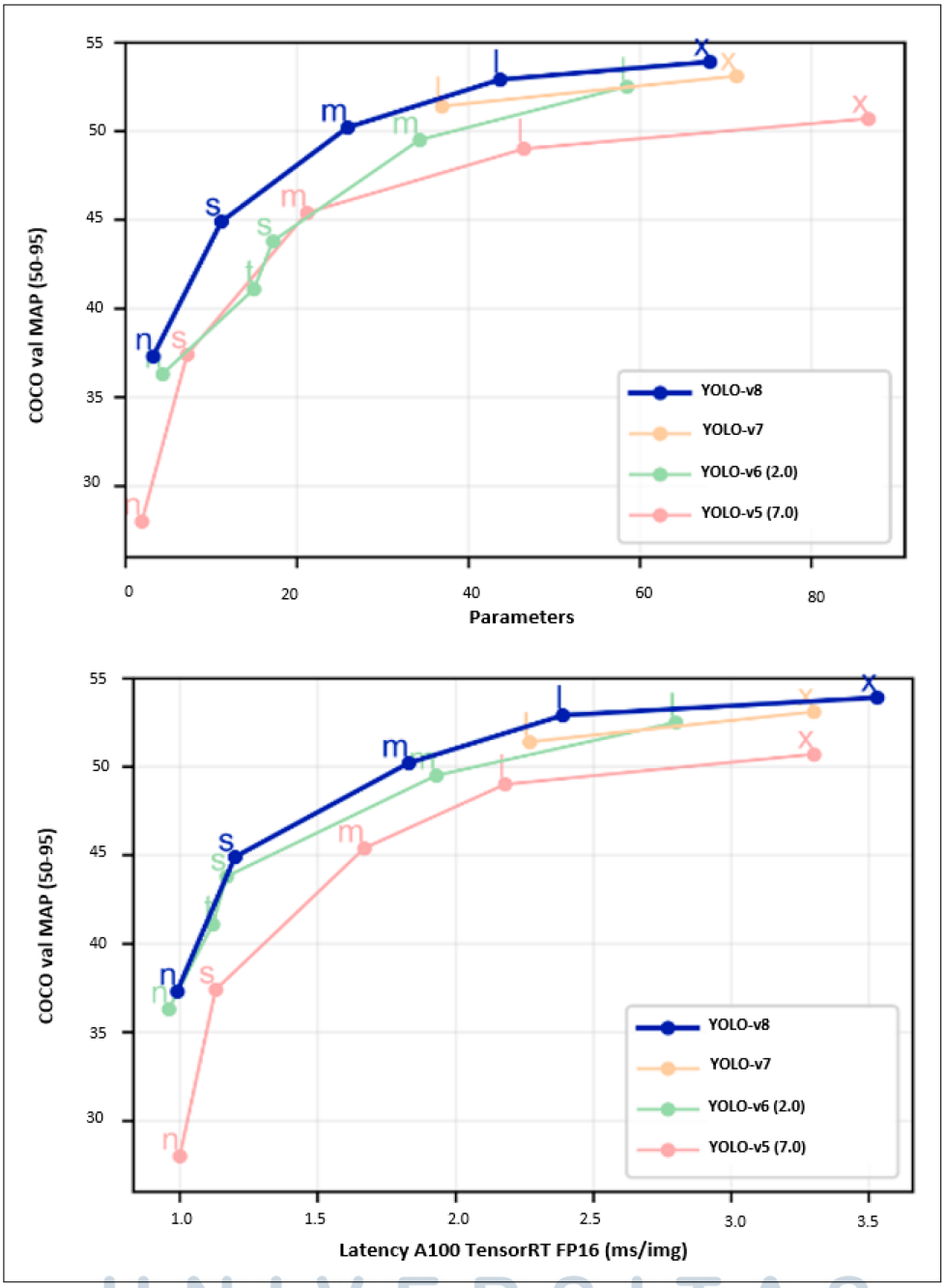
D Fully Connected Layer

Fully connected layer sepenuhnya terhubung menerima input dari lapisan output pooling atau konvolusi terakhir, yang diratakan sebelum disampaikan sebagai input. Proses meratakan output melibatkan menggulung semua nilai

dari output yang diperoleh setelah lapisan pooling atau konvolusi terakhir menjadi vektor (matriks 3D). Penambahan lapisan FC merupakan teknik sederhana untuk mempelajari kombinasi nonlinear dari fitur-fitur tingkat tinggi yang direpresentasikan oleh output lapisan konvolusi. Dalam ruang ini, lapisan FC belajar suatu fungsi mungkin yang nonlinear [19].

2.1.4 YOLOv8

YOLO, singkatan dari You Only Look Once, telah berkembang pesat sejak pertama kali diperkenalkan pada tahun 2015, mencapai puncaknya dengan rilis terbaru, YOLO-v8, pada Januari 2023. Variasi YOLO memprioritaskan kinerja real-time dan akurasi klasifikasi tinggi sambil beroperasi dalam batasan komputasi yang efisien. Seiring berjalannya waktu, model YOLO semakin menekankan prinsip-prinsip ini, berevolusi untuk memenuhi tuntutan berbagai aplikasi, termasuk inspeksi kualitas otomatis dalam deteksi cacat permukaan industri. Penambahan terbaru ke keluarga YOLO dikonfirmasi pada Januari 2023 dengan dirilisnya YOLO-v8 oleh Ultralytics (juga merilis YOLO-v5). Gambar 2.2 menunjukkan bahwa ketika membandingkan YOLO-v8 dengan YOLO-v5 dan YOLO-v6 yang dilatih pada resolusi gambar 640, semua varian YOLO-v8 menghasilkan throughput yang lebih baik dengan jumlah parameter yang sama, yang menunjukkan reformasi arsitektur yang hemat perangkat keras. Fakta bahwa YOLO-v8 dan YOLO-v5 disajikan oleh Ultralytics dengan YOLO-v5 memberikan kinerja real-time yang mengesankan dan berdasarkan hasil benchmarking awal yang dirilis oleh Ultralytics, diasumsikan kuat bahwa YOLO-v8 akan berfokus pada edge yang dibatasi penyebaran perangkat dengan kecepatan inferensi tinggi [21].



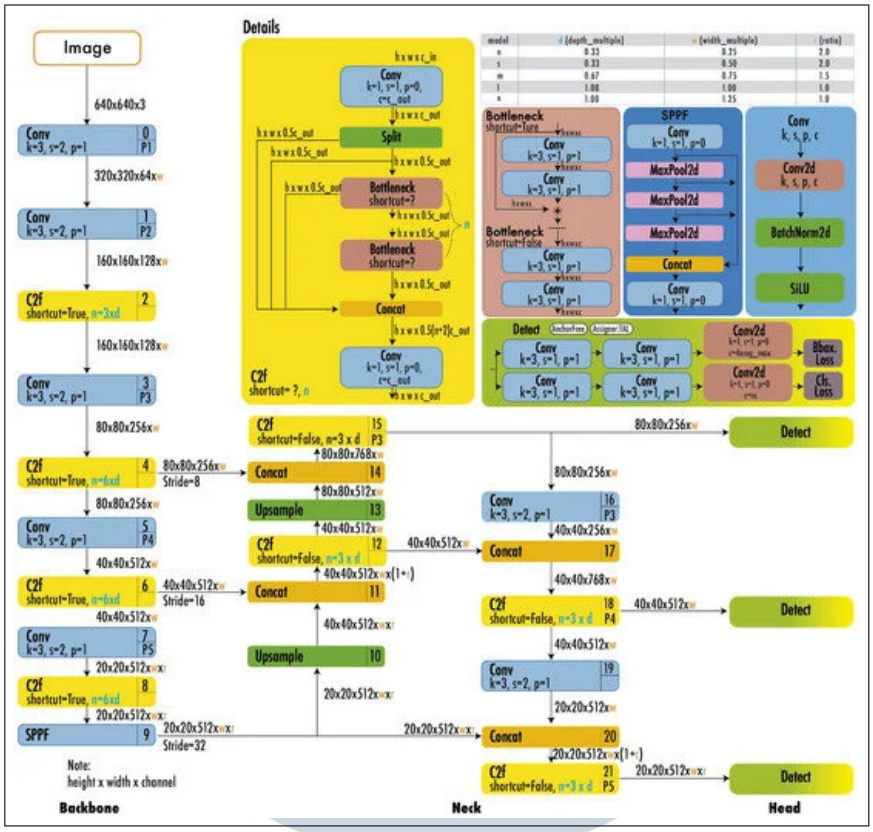
Gambar 2.2. Perbandingan YOLO-v8 dengan pendahulunya [21]

Model YOLO (You Only Look Once) telah mengalami serangkaian evolusi signifikan sejak versi pertamanya pada tahun 2016. Dimulai dengan YOLOv1 yang memperkenalkan deteksi objek *real-time* dengan satu kali lewat melalui jaringan saraf, setiap iterasi berikutnya seperti YOLOv2, v3, dan v4 terus meningkatkan akurasi dan kecepatan deteksi dengan menggunakan teknik-teknik

seperti normalisasi batch, jaringan piramida fitur, dan koneksi Cross-Stage Partial. YOLOv5 menggabungkan penjadwalan laju belajar otomatis, pelatihan presisi campuran, dan teknik pelatihan lainnya yang membuat pengimplementasiannya lebih mudah. YOLOv6 dan v7 terus memperbaiki fungsi aktivasi dan strategi augmentasi data untuk meningkatkan efisiensi dan akurasi deteksi objek. Versi terbaru, YOLOv8, ditujukan untuk aplikasi industri, khususnya dalam deteksi cacat, dengan fokus pada menjaga kecepatan inferensi tinggi dan meningkatkan akurasi, sesuai dengan permintaan inspeksi kualitas otomatis dalam industri manufaktur [21].

Arsitektur YOLOv8 menggunakan *backbone* yang serupa dengan YOLOv5 dengan beberapa perubahan pada CSPLayer, sekarang disebut modul C2f yang menggunakan model dari CNN. Modul C2f (cross-stage partial bottleneck dengan dua konvolusi) menggabungkan fitur tingkat tinggi dengan informasi kontekstual untuk meningkatkan akurasi deteksi. *Backbone* menghasilkan feature maps dengan resolusi berbeda. YOLOv8 menggunakan model tanpa anchor dengan kepala terpisah untuk memproses tugas keberadaan objek, klasifikasi, dan regresi secara independen. Desain ini memungkinkan setiap cabang untuk fokus pada tugasnya dan meningkatkan akurasi keseluruhan model. Pada lapisan output YOLOv8, YOLOv8 menggunakan fungsi sigmoid sebagai fungsi aktivasi untuk skor keberadaan objek, yang mewakili probabilitas bahwa kotak pembatas berisi objek. Ini menggunakan fungsi softmax untuk probabilitas kelas, yang mewakili probabilitas objek milik setiap kelas yang mungkin.

YOLOv8 menggunakan fungsi kerugian CIoU dan DFL untuk kerugian kotak pembatas dan entropi silang biner untuk kerugian klasifikasi. CIoU atau *Complete IoU* adalah sebuah metode yang diusulkan untuk regresi kotak pembatas dengan mempertimbangkan tiga ukuran geometris: area tumpang tindih, jarak titik tengah, dan rasio aspek [22]. DFL (Distribution Focal Loss) adalah varian dari Generalized Focal Loss (GFL), yang memperluas Focal Loss (FL) asli dari versi diskrit menjadi versi kontinu. GFL memungkinkan solusi yang dioptimalkan secara global untuk menargetkan nilai kontinu yang diinginkan, bukan hanya nilai diskrit. DFL, sebagai bagian dari GFL, membuat jaringan fokus pada pembelajaran probabilitas nilai di sekitar lokasi kontinu kotak pembatas target di bawah distribusi yang sewenang-wenang dan fleksibel [23]. Kerugian-kerugian ini telah meningkatkan kinerja deteksi objek, terutama saat menangani objek-objek kecil. Untuk rangkain arsitektur YOLOv8 dapat dilihat pada Gambar 2.3 [24].



Gambar 2.3. Arsitektur YOLOv8

[24]

Jadi pada arsitektur YOLOv8, *backbone* adalah tahap awal dalam arsitektur YOLOv8, yang bertanggung jawab untuk mengekstrak fitur dari gambar input. Pada YOLOv8, backbone menggunakan model CNN (Convolutional Neural Network), yaitu CSPDarknet53. CSP ini bertugas untuk membagi *feature maps* menjadi dua, yang pertama untuk layer konvulasi dan yang kedua untuk digabungkan dengan hasil konvulasi. *Backbone* menghasilkan feature maps dengan resolusi berbeda, yang kemudian digunakan oleh tahap selanjutnya. *Neck* berfungsi untuk menggabungkan *feature maps* dari *backbone* dengan resolusi berbeda. Bagian *head* berfungsi untuk mendeteksi objek yang terbagi menjadi tiga cabang, cabang pertama untuk mendeteksi objek kecil, kedua untuk ukuran sedang, dan yang terakhir untuk objek yang ukuran besar [22, 24].

2.1.5 Confusion Matrix

Confusion matrix adalah alat penting dalam klasifikasi terpandu dalam pembelajaran mesin, yang digunakan untuk mengevaluasi perilaku dan kinerja

model klasifikasi. Biasanya direpresentasikan sebagai struktur persegi, terdiri dari baris dan kolom, di mana baris mewakili kelas aktual dari contoh, dan kolom mewakili kelas yang diprediksi. Dalam klasifikasi biner, matriks kebingungan mengambil bentuk matriks 2x2, sedangkan untuk masalah multi-kelas, itu berkembang menjadi matriks k x k, di mana k mewakili jumlah kelas.

Matriks ini secara luas digunakan untuk menilai kinerja pengklasifikasi pada dataset. Di ranah rekayasa perangkat lunak, Font dkk. menerapkan matriks kebingungan untuk membedakan antara nilai-nilai yang diprediksi dan nilai-nilai sebenarnya dari elemen model. Matriks-Matriks ini khususnya menggunakan empat ukuran kunci dari matriks kebingungan - true positive (TP), false positive (FP), true negative (TN), dan false negative (FN) dapat dilihat pada Tabel 2.1 [25, 26]. Terdapat beberapa indikator performa yang dapat dihasilkan dari true positive (TP), false positive (FP), true negative (TN), dan false negative (FN), seperti *precision, recall*, dan *F-score* [27].

- *Precision* adalah ukuran keakuratan asalkan kelas tertentu telah diprediksi dapat dilihat pada Persamaan 2.1.

$$\text{Precision} = \frac{\#TP}{\#TP + \#FP} \tag{2.1}$$

- *Recall* adalah ukuran kemampuan model prediksi untuk memilih instance kelas tertentu dari kumpulan data dapat dilihat pada Persamaan 2.2 [27, 28].

$$\text{Recall} = \frac{\#TP}{\#TP + \#FN} \tag{2.2}$$

- *F1-score* adalah rata-rata harmonik antara *precision* dan *recall* dapat dilihat pada Persamaan 2.3 [27, 28].

$$\text{F1-score} = \frac{2xPrecisionxRecall}{Precision + Recall} \tag{2.3}$$

Tabel 2.1. Confusion Matrix

		Predicted Class	
		P	N
Actual Class	P	True Positive	False Negative
	N	False Positive	True Negative

2.1.6 mAP (mean Average Precision)

mAP (*mean Average Precision*) adalah metrik yang digunakan untuk mengukur akurasi detektor objek di seluruh kelas dalam suatu basis data tertentu. mAP dihitung dengan cara mengambil rata-rata dari nilai Average Precision (AP) yang didapatkan untuk setiap kelas individual. Average Precision (AP) mengukur area di bawah kurva presisi-recall untuk suatu kelas tertentu, dan dapat dilihat pada Persamaan 2.4.

$$AP_{\text{all}} = \sum_{n} (R_{n+1} - R_n) \cdot P_{\text{interp}}(R_{n+1}) \tag{2.4}$$

Di mana n adalah indeks iterasi saat ini, R_{n+1} adalah nilai *recall* iterasi berikutnya, R_n adalah nilai *recall* iterasi saat ini, dan $P_{\text{interp}}(R_{n+1})$ adalah presisi interpolasi yang dapat dilihat pada Persamaan 2.5.

$$P_{\text{interp}}(R_{n+1}) = \max_{\tilde{R}:\tilde{R} \geq R_{n+1}} P(\tilde{R}) \tag{2.5}$$

Di mana $\widetilde{R}$ adalah nilai dari recall, $\widetilde{R} \geq R_{n+1}$ adalah kondisi nilai recall lebih besar atau sama dengan nilai presisi saat ini, dan $P(\widetilde{R})$ adalah nilai presisi untuk level recall $\widetilde{R}$ tertentu. Jadi, *mean Average Precision* (mAP) didapat dari rata-rata AP pada semua kelas yang dapat dilihat pada Persamaan 2.6 [29].

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{2.6}$$

Dengan AP_i adalah nilai AP iterasi ke i, dan N adalah jumlah kelas yang dievaluasi.

Pada YOLOv8, terdapat dua jenis mAP yang umum digunakan, yaitu mAP@0.5 dan mAP@0.5:0.95. mAP@0.5 mengacu pada nilai mAP yang dihitung dengan menggunakan ambang IoU sebesar 0.5, metriks ini mengukur kinerja deteksi secara keseluruhan [30] . Ketika nilai mAP mencapai atau melebihi 0.5,

prediksi dianggap benar. Sementara itu, mAP@0.5:0.95 menghitung mAP dalam rentang 0.5 hingga 0.95 dengan langkah sebesar 0.05 [31]. IoU (*Intersection over Union*) adalah metrik evaluasi standar dalam deteksi objek yang digunakan untuk menentukan true positives dan false positives dalam sekumpulan prediksi. Penting untuk memilih ambang akurasi yang sesuai saat menggunakan IoU sebagai metrik evaluasi [32].

2.1.7 Evaluasi Model

Evaluasi dapat dilakukan dengan memperhatikan *Losses*, *Precision, Recall, and F1-score*, dan mAP (mean Average Precision) yang dihasilkan dari proses pelatihan menggunakan YOLOv8. Evaluasi dapat dilakukan melalui matriks gambaran dan statistik yang dihasilkan dari proses pelatihan menggunakan YOLOv8. Matriks dan statistik yang diperoleh meliputi:

- *Losses*: Menunjukkan kerugian selama pelatihan membantu dalam memahami seberapa baik model belajar. Penurunan kerugian menunjukkan peningkatan kemampuan model dalam mendeteksi dan mengklasifikasikan objek dengan akurat [33].
- *Confusion Matrix*: Digunakan untuk mengukur akurasi model dengan membandingkan nilai yang diprediksi dengan nilai yang sebenarnya [34].
- *Precision, Recall, and F1-score*: Metrik-metrik ini memberikan pemahaman yang lebih detail tentang kinerja model Anda pada setiap kelas. Presisi yang tinggi menunjukkan lebih sedikit positif palsu, *recall* yang tinggi menunjukkan lebih sedikit negatif palsu, dan nilai F1 yang tinggi menandakan keseimbangan yang baik antara presisi dan *recall* [35].
- mAP (mean Average Precision): mAP adalah metrik yang banyak digunakan untuk tugas deteksi objek. Ini memberikan ukuran keseluruhan dari kinerja model Anda di semua kelas dan sangat berguna untuk membandingkan model-model berbeda atau melacak kinerja dari waktu ke waktu [36].