BAB II

LANDASAN TEORI

2.1. Fast-Moving Consumer Goods (FMCG)

Fast-Moving Consumer Goods (FMCG) adalah produk-produk yang dapat terjual secara cepat dengan harga yang murah, biasanya merupakan kebutuhan sehari-hari, sebagai contoh yaitu seperti minuman dan makanan ringan, produk perawatan tubuh, atau barang kelontong (Ar & Kurniawan, 2021). Kategori ini terutama makanan dan minuman dipilih karena sering dibeli dan umum ditemukan di pasar, ritel, minimarket, dan supermarket. Selain itu, secara tampilan memiliki ciri visual yang mudah dipelajari model, sehingga banyak dataset barang ritel jenis ini beredar dan digunakan pada penelitian.

2.2. Self Checkout System

Sesuai dengan namanya, *self checkout system* adalah pendekatan modern dalam proses pembayaran toko yang memungkinkan pelanggan untuk memindai dan membayar barang secara mandiri tanpa bantuan kasir. Dengan pendekatan ini, proses transaksi menjadi lebih cepat dibanding metode konvensional, di mana kasir harus memindai setiap barang secara manual, dan mengakibatkan antrian panjang yang sering menjadi tantangan besar di toko ritel (Gazzola dkk., 2022). Menurut (Hamir dkk., 2024), contoh keberhasilan yang didapatkan dengan penerapan sistem checkout otomatis adalah pada kasus Amazon Go. Mereka menghapus keperluan konter pembayaran

konvensional dengan menggantinya menggunakan sistem yang terintegrasi *computer vision* dan sensor fusi, nantinya ketika pembeli keluar dari toko, semua barang yang sudah dibeli secara otomatis muncul pada tagihan di dalam aplikasi, terbukti efektif menghilangkan antrian dan mengurangi waktu *checkout* dan pembayaran.

2.3. Deep Learning

Deep learning atau pembelajaran mendalam merupakan salah satu cabang dari lingkungan *machine learning* atau pembelajaran mesin yang menggunakan abstraksi pemodelan tingkat tinggi untuk mempelajari representasi data dengan serangkaian fungsi non-linier (Zahir & Adi Saputra, 2024).

2.4. Transfer Learning

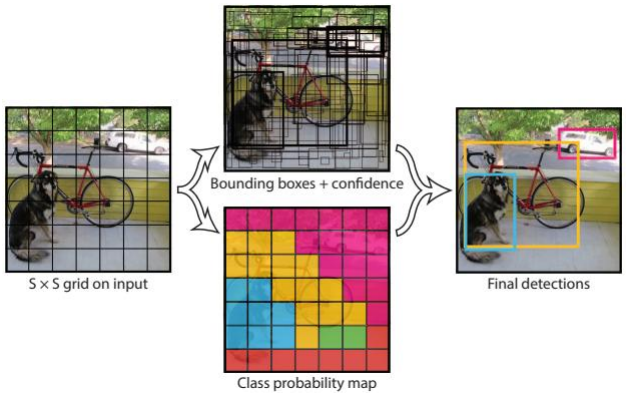
Transfer learning merupakan salah satu pendekatan dari *deep learning* dengan memanfaatkan model jaringan saraf yang telah dilatih sebelumnya dan digunakan sebagai awalan untuk mempelajari tugas atau masalah yang beda (Faturrahman dkk., 2023).

2.5. You Only Look Once (YOLO)

You Only Look Once atau populernya disingkat YOLO merupakan pendekatan dalam pendeteksian objek dengan memisahkan spasial bounding box dan probabilitas kelas (Andaru, Imam, 2024). Model YOLO merupakan *one-stage detection* yang deteksinya langsung melakukan deteksi objek (termasuk klasifikasi dan *bounding box*) dalam satu langkah (Redmon &

Farhadi, 2017). Metode *one-stage detection* secara general mempunyai kecepatan yang jauh lebih tinggi daripada *two-stage detection* sehingga membuatnya lebih sesuai untuk kasus *real-time detection* (Oh dkk., 2022).

Pada penelitian ini menggunakan *pre-trained model* YOLO, sehingga termasuk ke dalam kategori *Transfer Learning*. *Pre-trained model* YOLO menggunakan jaringan saraf yang sudah dilatih dengan dataset besar seperti ImageNet, COCO, dan VOC sehingga jaringan konvolusi di dalamnya sudah mempunyai kemampuan yang tinggi dalam mempelajari fitur informasi dalam data citra (Situ dkk., 2023).



Gambar 2.5.1 Proses Deteksi YOLO

Sumber : (Redmon dkk., 2016)

Pada jaringan YOLO pemrosesan gambar bisa dilihat pada Gambar 2.5.1, input gambar diproses menjadi grid berukuran S x S, setiap grid memprediksi kotak pembatas dan menghasilkan skor kepercayaan (*Confidence Score*) pada masing-masing kotak tersebut yang nantinya skor ini mencerminkan seberapa yakin model bahwa area kotak tersebut berisi objek dan seberapa akurat kotak tersebut diprediksi (Redmon dkk., 2016). Pada prosesnya, model YOLO

menggunakan metode *Intersection over Union* (IoU) dalam menghasilkan nilai kepercayaannya, digunakan perhitungan dengan Persamaan (2.1).

$$IoU\ (A,B) = \frac{A \cap B}{A \cup B}; IoU\ (A,B) \in [0,1] \qquad (2.1)$$

Keterangan :

$IoU\ (A,B)$: Fungsi *Intersection over Union* untuk gambar prediksi dan label

A : Prediksi

B : Label

Pada model YOLO juga terdapat ambang batas (*Threshold*) IoU, sebagai patokan apakah deteksi objek berhasil atau tidak, jika skor kepercayaan melebihi ambang batas, maka deteksi dianggap berhasil. Selain itu ambang batas meminimalisir multi *bounding box* dan mengambil box dengan nilai IoU yang terbesar untuk ditampilkan (Redmon dkk., 2016).

Masing-masing kelas objek dihitung rata-rata akurasinya (*Average Precision*) dengan Persamaan (2.2).

$$AP = \frac{TP}{TP+FP} \quad (2.2)$$

Keterangan :

AP : *Average Precision*

TP : *True Positive*

FP : *False Positive*

True Positive adalah jumlah prediksi positif dan *False Positive* adalah jumlah prediksi yang dianggap positif namun sebenarnya negatif. Jika AP = 1 artinya tingkat deteksi sempurna, sedangkan jika AP = 0 maka hasil deteksinya buruk (Andaru, Imam, 2024). Pada keseluruhannya, proses evaluasi

keseluruhan *Average Precision* atas semua objek dalam *dataset* menggunakan metrik *mean average precision* (mAP) dengan Persamaan (2.3)

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k \quad (2.3)$$

Keterangan :

mAP : Rata-rata dari *Average Precision*

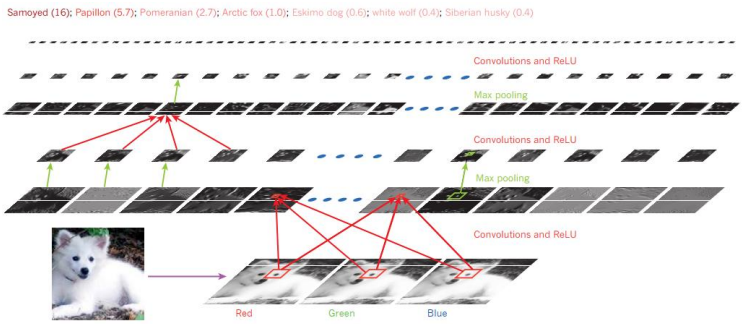
k : Jumlah kelas

AP_k : *Average Precision* untuk kelas k

Perusahaan Ultralytics selaku pembuat dari model YOLOv5 dan YOLOv8 telah mengeluarkan versi YOLO terbarunya di bulan September 2024, yaitu YOLOv11. Pada dokumentasi resminya, mereka mengklaim bahwa versi ini jauh lebih baik daripada versi-versi YOLO sebelumnya, baik dari segi kecepatan, efisiensi, akurasi, dan parameter model yang lebih sedikit yang membuat ukurannya lebih kecil serte fleksibel untuk di deploy pada berbagai lingkungan pengembangan (Ultralytics, 2024).

2.6. Convolutional Neural Network (CNN)

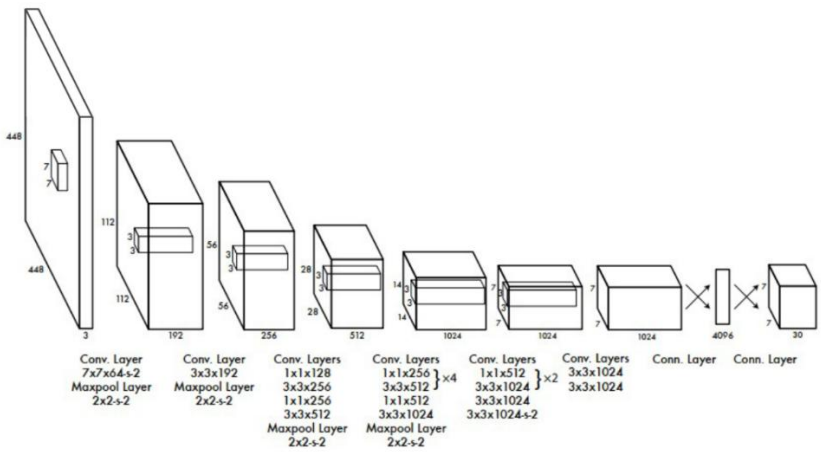
Convolutional Neural Network (CNN) adalah salah satu algoritma jaringan saraf tiruan yang umum digunakan untuk klasifikasi gambar (Winiarti dkk., 2022). Proses konvolusi ini digunakan dalam arsitektur YOLO, diterapkan secara berulang melalui beberapa laporan konvolusi untuk mendeteksi pola-pola penting pada gambar, selain itu CNN juga dipakai untuk memprediksi bounding box dan probabilitas kelas objek secara langsung (Diwan dkk., 2023).



Gambar 2.6.1 Proses Konvolusi

Sumber : (Lecun dkk., 2015)

Proses konvolusi divisualisasikan pada Gambar 2.6.1, menurut Lecun dkk., (2015) CNN memecah gambar menjadi patch kecil dengan filter lalu mendeteksi pola dengan lapisan yang berlapis-lapis di dalamnya. Lapisan awal berfungsi untuk mendeteksi pola sederhana, seperti tepi atau garis. Lapisan tengah mendeteksi kombinasi dari pola-pola sederhana seperti sudut atau tekstur, dan lapisan akhir menggabungkan pola-pola kompleks untuk mendeteksi objek secara lengkap.



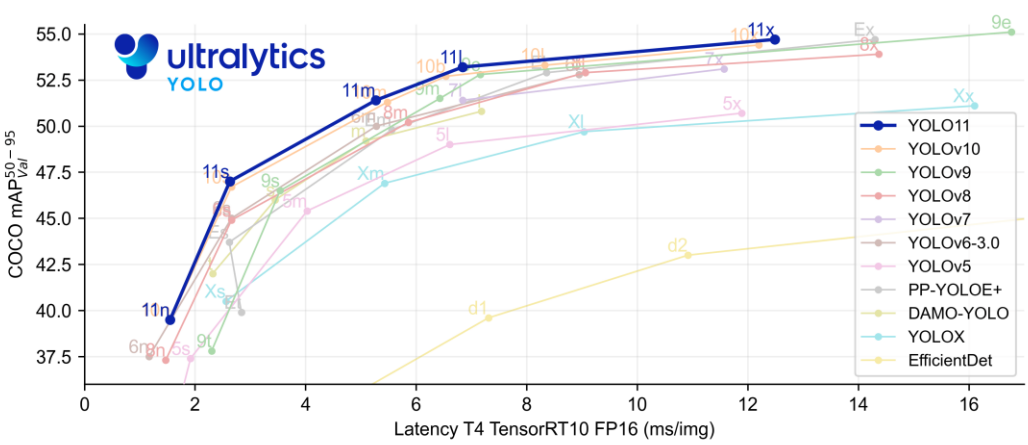
Gambar 2.6.2 Lapisan Konvolusi pada YOLO

Sumber : (Redmon dkk., 2016)

CNN digunakan berlapis-lapis pada YOLO seperti pada Gambar 2.6.2, dimana CNN pada YOLO langsung belajar untuk menghubungkan fitur citra dengan keluaran yang dibutuhkan, memungkinkan YOLO untuk memproses seluruh gambar dalam satu kali deteksi melalui arsitektur jaringan di dalamnya, sehingga membuatnya cepat dalam mendeteksi objek dan cocok untuk penggunaan *real-time*.

2.7. YOLOv11

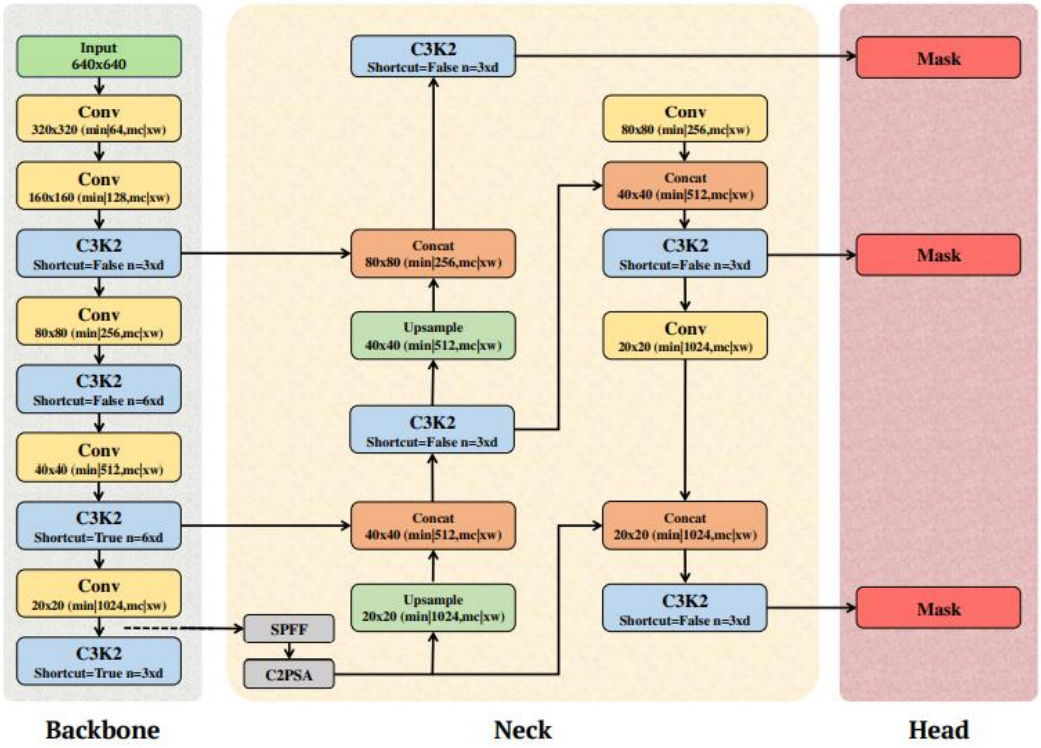
YOLOv11 merupakan versi terbaru dari seri YOLO yang dikeluarkan oleh perusahaan Ultralytics yang membuat YOLOv5 dan YOLOv8, versi terbaik yang banyak digunakan sebelumnya. Pada dokumentasi resminya, YOLOv11 ini menunjukkan banyak peningkatan sehingga lebih baik dari segi akurasi, kecepatan dan efisiensinya (Ultralytics, 2024).



Gambar 2.7.1 Perbandingan *Benchmark* YOLOv11 dengan versi sebelumnya

Sumber : (Ultralytics, 2024)

Berdasarkan Gambar 2.7.1, hasil perbandingan akurasi model YOLO dengan waktu inferensi menggunakan akselerator AI TensorRT10 menunjukkan bahwa grafik YOLOv11 berada di posisi teratas dibandingkan YOLO lainnya. Hal ini mengindikasikan bahwa YOLOv11 memiliki keunggulan baik dari segi kecepatan inferensi maupun akurasi dibandingkan versi pendahulunya.

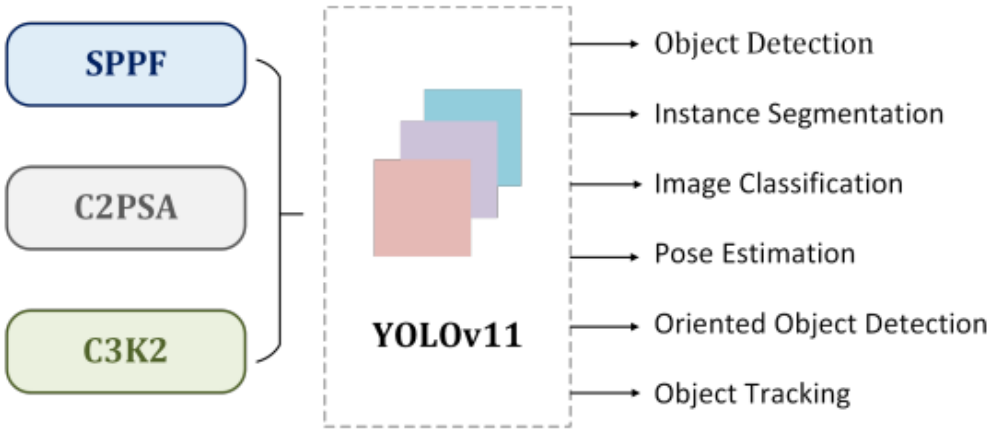


Gambar 2.7.2 Arsitektur YOLOv11

Sumber : (Khan & Jensen, 2024)

Arsitektur YOLOv11 secara keseluruhan bisa dilihat pada Gambar 2.7.2, disana disajikan diagram skema yang menggambarkan tiga komponen inti dari YOLOv11. Menurut Khan & Jensen, (2024), arsitektur YOLOv11 memiliki beberapa komponen yaitu *Backbone*, *Neck* dan *Head*. *Backbone* menangani

ekstraksi fitur *multi-scale* dengan blok tingkat lanjut seperti C3K2 dan *Spatial Pyramid Pooling Fast* (SPPF) yang dirancang untuk representasi fitur yang efisien. Bagian *Neck* menyempurnakan fitur ini dengan mekanisme tambahan seperti *Cross-Stage Partial with Attention* (C2PSA). Terakhir, Head memprediksi *bounding box* dari objek, *mask*, dan klasifikasinya dengan *multi-scale processing* yang ditingkatkan.



Gambar 2.7.3 Blok Penting dari YOLOv11 dan Fungsinya

Sumber : (Khanam & Hussain, 2024)

Menurut (Khanam & Hussain, 2024) YOLOv11 mempertahankan struktur yang serupa dengan pendahulunya, dengan memanfaatkan lapisan konvolusi awal yang membentuk dasar dari proses ekstraksi fitur, yang secara bertahap mengurangi dimensi spasial. Peningkatan yang dilakukan YOLOv11 adalah seperti pada Gambar 2.7.3, yaitu penggantian blok C2F yang digunakan pada versi sebelumnya untuk *upsampling* dan *concatenation*. Blok C3K2 lebih efisien secara komputasi. Tanda K2 di C3K2 menunjukkan ukuran *kernel* yang lebih kecil, sehingga berkotribusi lebih cepat dalam pemrosesan dengan

mempertahankan kinerja. Selain itu, pada YOLOv11 ditambahkan blok untuk *attention mechanism* dengan C2PSA, blok ini memungkinkan model untuk konsentrasi pada area gambar objek yang membuatnya lebih akurat dalam mendeteksi objek dibanding dengan YOLOv8 yang lebih lemah dalam *attention mechanism*.

2.8. Hyperparameter

Hyperparameter adalah nilai-nilai parameter yang ditetapkan sebelum proses pembelajaran dimulai untuk meningkatkan kualitas deteksinya sesuai dengan model yang dipakai sehingga konfigurasinya memiliki dampak yang signifikan pada kinerja dan model yang dihasilkan (T. A. E. Putri dkk., 2023). Menurut (Chhabra & Goyal, 2024) konfigurasi *Hyperparameter* dibutuhkan pada kasus ini dimana objek ritel seringkali berukuran kecil atau bentuk dan rupa yang mirip antara satu produk dengan produk lainnya sehingga model harus mempelajari secara lebih detail dari masing-masing citra objek agar mendeteksi secara akurat

2.9. Optimizer

Optimizer adalah algoritma atau metode dalam kecerdasan buatan untuk menyesuaikan parameter seperti bobot dan bias, dengan tujuan mengurangi fungsi kerugian atau memfasilitasi perubahan nilai bobot dan penyesuaian laju pelatihan dalam jaringan saraf agar kerugian dapat diminimalkan (Ayu & Pradipta, 2022). *Optimizer* ini termasuk ke dalam *hyperparameter* yang dapat

disesuaikan nilainya sebelum menjalankan proses pelatihan pelatihan ulang (*fine-tuning*).

2.10. Adam Weight Decay

Optimizer AdamW adalah variasi dari optimizer Adam yang memperkenalkan teknik weight decay secara lebih eksplisit dalam proses optimasi dan memodifikasi Adam dengan memisahkan *weight decay* dari proses *update* parameter (Mahajaya dkk., 2024). Formula yang dipakai tertera pada Persamaan 2.4.

$$\theta_{t+1} = \theta_t - \frac{\alpha \cdot \hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} - \alpha\, \omega d\, \theta_t \quad (2.4)$$

Keterangan :

θ_t : Parameter model pada iterasi ke-t

α : *Learning rate*

$\hat{m}_t$: Estimasi *bias-corrected* dari *momentum* (mean dari gradien)

$\hat{v}_t$: Estimasi *bias-corrected* dari *second momentum* (varians gradien)

ϵ : Bilangan kecil untuk mencegah pembagian dengan nol

ω : Koefisien *weight decay* (konstanta regularisasi)

$\alpha\, \omega d\, \theta_t$: Bagian *decoupled weight decay* dari AdamW

2.11. Mixup Augmentation

Mixup Augmentation merupakan salah satu teknik augmentasi data dengan memadukan dua gambar dan labelnya ke dalam satu gambar, Teknik ini membuat model lebih tahan terhadap variasi data yang tidak terduga atau *noise*

(Zhang dkk., 2021). Augmentasi data juga termasuk ke dalam *hyperparameter* yang dapat ditambahkan sebelum proses pelatihan ulang (*fine-tuning*).

2.12. Open Neural Network Exchange (ONNX)

Open Neural Network Exchange (ONNX) adalah format standar yang dirancang untuk mempresentasikan model *machine learning* dalam bentuk graf yang menampilkan langkah-langkah proses komputasi secara matematis, format ini mendefinisikan semua operasi yang diperlukan model, termasuk transformasi fitur menjadi prediksi, sehingga memungkinkan modelnya digunakan pada lintas framework dan platform (ONNX, 2024). Pada *library* ONNX juga terdapat fitur untuk kuantisasi model, banyak opsi kuantisasi yang disediakan di dalamnya sehingga sangat mudah untuk dilakukan.

2.13. Kuantisasi Model

Kuantisasi model merupakan salah satu cara yang dapat dilakukan untuk mengurangi waktu inferensi saat menggunakan jaringan saraf, dengan mengubah format data yang diproses ke bentuk bit yang lebih rendah agar lebih mudah diproses oleh mesin (Nagel dkk., 2021). Contohnya pada penelitian ini mengubah pemrosesan data model dari Float 16 (FP16) (default YOLOv11) ke Integer 8 (INT8) Namun, kuantisasi model memiliki kekurangan, karena memperbanyak *noise* atau *bias* ke dalam jaringan sarafnya yang dapat menyebabkan penurunan akurasi, walaupun beberapa jaringan tangguh pada pengaruh noise atau bias (Andaru, Imam, 2024).

Terdapat dua jenis metode kuantisasi, yaitu kuantisasi statis dan kuantisasi dinamis. Menurut dokumentasi resmi dari (ONNX, 2024), kuantisasi dinamis menghitung parameter kuantisasi, seperti *scale* dan *zero point*, secara *real-time* selama inferensi, sehingga mengurangi waktu inferensi dan mempertahankan akurasi yang lebih tinggi sekaligus dalam satu waktu dibandingkan dengan kuantisasi statis. Sebaliknya, kuantisasi statis menghitung parameter kuantisasi di awal dengan menggunakan data kalibrasi untuk *scale* dan *zero point* nya, lalu parameter tersebut disimpan untuk semua proses inferensi. Kuantisasi statis cocok untuk kasus yang membutuhkan optimasi kecepatan inferensi secara maksimum, meskipun akurasinya lebih rendah.

Secara default, ONNX Runtime menggunakan *asymmetric quantization* untuk kuantisasi bertipe dinamis.

$$zero\ point = round\left(-\frac{data\_range\_min}{scale}\right) + quantization\_range\_min \quad (2.5)$$

$$scale = \frac{data\_range\_\max - data\_range\_min}{quantization\_range\_\max - quantization\_range\_min} \quad (2.6)$$

$$val_{quantized} = round\left(\frac{val_{fp16} - zero\ point}{scale}\right) \quad (2.7)$$

Perhitungan *asymmetric quantization* menggunakan Persamaan (2.5), (2.6), dan (2.7) karena menggunakan kuantisasi dinamis, maka nilai *zero point* dan *scale* ditentukan secara dinamis oleh ONNX Runtime dengan melihat distribusi angka pada dataset yang diproses untuk ditetapkan sebagai data_range_max dan data_range_min. Selain itu, dikarenakan kasus pada penelitian ini menggunakan gambar, dan YOLO memproses gambar secara RGB, maka tipe kuantisasi yang dipakai adalah *unsigned* INT8 atau UINT8. UINT8 memiliki

rentang angka dari 0 sampai 255 sehingga cocok dipakai untuk mengolah data gambar RGB yang nilainya tidak ada negatif karena sama-sama memiliki rentang secara integer dari 0 sampai 255 (Pérez-Delgado & Celebi, 2024).

Pada penelitian ini menggunakan metode *Post-Training Quantization* (PTQ) daripada menggunakan *Quantization-Aware Training* (QAT). PTQ lebih mendukung untuk deployment cepat dan ringan daripada QAT yang membutuhkan waktu dan sumber daya lebih untuk melakukan proses pelatihan ulang untuk mengubah arsitektur model secara keseluruhan, sehingga lebih membebani perangkat keras (Zhao dkk., 2023).

2.14. Confusion Matrix

Confusion Matrix adalah metode evaluasi yang digunakan untuk menilai kinerja model deep learning dalam pengklasifikasian objek dengan menunjukkan perbandingan prediksi model dan data asli yang digunakan dalam pengujian (Nur dkk., 2024). *Confusion Matrix* memiliki beberapa artibut yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Metode ini memungkinkan perhitungan berbagai metrik evalusasi seperti akurasi, *precision*, *recall*, dan *F1-score* sehingga memberikan gambaran mendalam terkait performa klasifikasi pada dataset tertentu (Cahyanti dkk., 2020) (Novita dkk., 2021). Gambaran dari metode *Confusion Matrix* bisa dilihat pada Tabel 2.14.1 berikut.

Tabel 2.14.1 Confusion Matrix

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

True Negative merupakan nilai hasil dari banyak data yang tepat dengan output salah. *True Positive* adalah banyak data yang berhasil diprediksi sesuai labelnya. *False Positive* adalah banyak data hasil prediksi yang salah diidentifikasi sebagai benar. Dan *False Negative* adalah banyak data yang dikategorikan negatif atau salah (N. H. Putri dkk., 2023). Formula yang digunakan dalam perhitungan *Confusion Matrix* terdapat pada Persamaan (2.7), (2.8), (2.9) dan (2.10).

$$\text{Accuracy (\%)} = \frac{(TP+TN)}{(TP+FP+TN+FN)} \quad (2.7)$$

$$\text{Precision (\%)} = \frac{TP}{(FP+TP)} \quad (2.8)$$

$$\text{Recall (\%)} = \frac{(TP+TN)}{(TP+FP+TN+FN)} \quad (2.9)$$

$$\text{F1-score (\%)} = \frac{(2\ x\ recall\ x\ precision)}{(recall+Precision)} \quad (2.10)$$

2.15. Penelitian Terkait

Penelitian sebelumnya berfungsi untuk menganalisa dan membedakannya dengan penelitian yang akan dilakukan. Dalam penelitian ini disertakan lima

jurnal penelitian sebelumnya terkait penggunaan model YOLO untuk objek ritel dan kuantisasi, antara lain seperti pada Tabel 2.15.1.

Tabel 2.15.1 Perbandingan Penelitian Sebelumnya (*State of the Art*)

No.	**Judul Jurnal dan Peneliti**	**Tahun dan Tempat Peneliti**	**Metode Penelitian**	**Objek Penelitian**	**Perbandingan yang dijadikan alasan tinjauan penelitian**
1.	*An Efficient Grocery Detection System Using HYOLO-NAS Deep Learning Model for Visually Impaired People.* (Chhabra & Goyal, 2024)	2024, India	YOLO-NAS + Kuantisasi + Tuning Hyperparameter	Grozi-120 (120 produk, total 4829 gambar), Roboflow Retail dataset (20 produk, total 2611 gambar)	Hasil penelitian ini memberikan wawasan penting mengenai rekomendasi *tuning hyperparameter* terutama pada penggunaan optimizer Adam Weight Decay dalam kasus deteksi produk ritel. Selain itu penelitian ini juga menunjukkan manfaat kuantisasi dalam mempercepat waktu inferensi. Juga melihat dari hasil

					dataset yang menggunakan roboflow lebih besar akurasinya, menjadikannya referensi dalam pemilihan format dataset yang dipakai
2.	*Retail Product Object Detection using YOLOv5 for Automatic Checkout System in Smart Retail Environment* (Hamir dkk., 2024)	2024, Malaysia	YOLOv5 + Setting Hyperparameter default COCO	RPC dataset, hanya mengambil produk yang ada di Malaysia (26 Produk, total 4160 gambar)	Berdasarkan penelitiannya, ditemukan bahwa penggunaan model YOLOv5 dengan pengaturan hyperparameter default untuk dataset COCO mungkin kurang optimal ketika diterapkan dalam deteksi objek ritel, hal ini dikarenakan peneliti menyimpulkan bahwa

					model mereka mengalami overfitting karena kinerjanya cukup buruk dalam mendeteksi gambar baru. Selain itu, penelitian ini juga memberikan referensi mengenai pemanfaatan platform roboflow dalam proses pelabelan dan fitur-fitur *pre-processing* di dalamnya untuk meningkatkan kualitas dataset yang digunakan dalam pelatihan model YOLO.
3.	*Analisis Kinerja Deteksi Gerakan dan Pengenalan*	2024, Indonesia	YOLOv8 + Setting	Dataset pribadi, dengan	Hasil dari penelitian ini menunjukkan bahwa model

	Objek Produk Ritel Berbasis YOLOv8 (Kusumawardhani dkk., 2024)		Hyperparameter default COCO	format VOC 2012 (10 produk, total 2383 gambar)	YOLOv8 cukup cocok dalam kasus deteksi produk ritel namun terjadi sedikit penurunan akurasi deteksi pada data baru atau data yang belum dilihat sebelumnya karena model terlalu memfokuskan detail-detail kecil atau *noise* pada data pelatihan (indikasi *overfitting*). Selain itu faktor barang yang bertumpuk juga menjadi tantangan untuk penelitian kedepannya.
4.	*Pengembangan Sistem Deteksi On-Shelf Availability*	2024, Indonesia	YOLOv8 + Kuantisasi ONNX	Dataset pribadi (135 produk	Penelitian ini memberikan wawasan terkait penerapan kuantisasi ke bit yang

	Produk Menggunakan Algoritma YOLOv8 pada Aplikasi Bergerak (Andaru & Fudholi, 2024)		+ Inferensi *Mobile*	dengan total 11216 gambar)	lebih kecil pada jaringan model, termasuk dampak positif dan negatifnya terutama pada perangkat *mobile*. Namun, peneliti pada jurnal ini belum mengeksplorasi lebih lanjut bagaimana efek kuantisasi jika diterapkan pada jenis perangkat lain di luar *mobile*. Walaupun model YOLOv8s dan YOLOv8n memberikan performa yang baik, peneliti juga merekomendasikan optimasi lebih lanjut pada hyperparameter guna meningkatkan

					akurasi dan kinerja deteksi secara keseluruhan dalam mendeteksi objek ritel.
5.	*Enhanced Self-Checkout System for Retail Based on Improved YOLOv10* (Tan dkk., 2024)	2024, Amerika Serikat	YOLOv10 yang dimodifikasi dengan *head* dari YOLOv8	RPC Dataset (200 produk dengan total 83000 gambar)	Penelitian ini menunjukkan bahwa YOLOv10 masih memiliki beberapa keterbatasan yang perlu diatasi melalui modifikasi, seperti yang dilakukan dalam jurnal ini. Temuan ini dapat menjadi dasar tinjauan untuk mengkaji apakah versi terbaru dari YOLO mampu memberikan peningkatan dalam mendeteksi objek ritel secara lebih akurat

					dan efisien atau tidak. Selain itu, mereka juga merekomendasikan variasi dataset berbagai sudut dalam rotasi objek 360° untuk mengatasi berbagai sudut pandang produk.

Berdasarkan temuan dari penelitian-penelitian yang telah dikaji, model YOLO telah banyak digunakan dalam deteksi objek ritel, tetapi tidak semua versinya sesuai untuk skenario ini. Versi yang lebih mendekati model terbaru, seperti YOLOv10, masih memiliki kelemahan yang memerlukan modifikasi arsitektur. Hal ini membuka peluang untuk mengevaluasi versi terbaru, yaitu YOLOv11, guna menentukan kelayakannya dalam mendeteksi objek ritel. Selain itu, beberapa penelitian menunjukkan bahwa konfigurasi hyperparameter yang tepat diperlukan untuk mengoptimalkan performa deteksi dalam skenario ini.

Di sisi lain, walaupun YOLOv11 terbilang sudah efisien untuk berbagai kasus, salah satu penelitian merekomendasikan penelitian selanjutnya untuk melakukan percobaan kuantisasi ke *bit* yang lebih rendah. Kuantisasi telah terbukti efektif dalam mengurangi ukuran model dan mempercepat waktu

inferensi, menjadikannya lebih sesuai untuk implementasi *real-time detection* untuk kedepannya. Namun, dampaknya terhadap akurasi masih menjadi tantangan, karena penurunan akurasi yang signifikan dapat meningkatkan risiko kesalahan deteksi dan berpotensi menyebabkan kerugian dalam sistem ritel otomatis. Oleh karena itu, penelitian ini berfokus pada pengembangan YOLOv11 dengan kuantisasi dan optimasi *hyperparameter* dalam mendeteksi objek ritel guna mencapai keseimbangan antara akurasi, ukuran model dan kecepatan inferensi.