PENGEMBANGAN MODEL DETEKSI UNTUK ON-SHELF AVAILABILITY PRODUK MENGGUNAKAN YOLOV8 PADA APLIKASI BERGERAK

Disusun Oleh:

N a m a : Gabriel Imam Andaru

NIM : 20523239

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2024

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGEMBANGAN MODEL DETEKSI UNTUK ON-SHELF AVAILABILITY PRODUK MENGGUNAKAN YOLOV8 PADA APLIKASI BERGERAK

TUGAS AKHIR

Disusun Oleh:

N a m a : Gabriel Imam Andaru

NIM : 20523239

Yogyakarta, 25 Juli 2024

Pembimbing,

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PENGESAHAN DOSEN PENGUJI

PENGEMBANGAN MODEL DETEKSI UNTUK ON-SHELF AVAILABILITY PRODUK MENGGUNAKAN YOLOV8 PADA APLIKASI BERGERAK

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 23 Juli 2024

Tim Penguji

Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. _____

Anggota 1

Ari Sujarwo, S.Kom., M.I.T. _____

Anggota 2

Erika Ramadhani, S.T., M.Eng. _____

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Gabriel Imam Andaru

NIM : 20523239

Tugas akhir dengan judul:

PENGEMBANGAN SISTEM DETEKSI ON-SHELF AVAILABILITY PRODUK MENGGUNAKAN YOLOV8 PADA APLIKASI BERGERAK

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 08 Juli 2024

(Gabriel Imam Andaru)

HALAMAN PERSEMBAHAN

Alhamdulillah, dengan penuh rasa syukur, penulis menyampaikan terima kasih atas segala dukungan dan kesempatan yang memungkinkan terselesaikannya tugas akhir ini, yang berjudul "PENGEMBANGAN MODEL DETEKSI UNTUK ON-SHELF AVAILABILITY PRODUK MENGGUNGAKAN YOLOV8 PADA APLIKASI BERGERAK". Penulis juga mengucapkan terima kasih yang sebesar-besarnya kepada Allah SWT yang telah menghadirkan orang-orang yang selalu memberikan semangat dan doa, sehingga tugas akhir ini dapat diselesaikan dengan baik. Tugas akhir ini penulis persembahkan untuk:

1. Bapak Bimo Nugroho, Ibu Sri Hardjanti dan Mikael Hanugrah Brahmantya selaku keluarga penulis yang telah memberikan dukungan secara emosional dan materiil.
2. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. selaku dosen pembimbing yang telah membantu penulis menyelesaikan tugas akhir ini dari awal hingga akhir.
3. Diri saya sendiri selaku penulis yang telah mencurahkan isi pikiran dan mampu menyelesaikan laporan ini pada waktu yang tepat.

HALAMAN MOTO

"Pengetahuan adalah kunci kesuksesan yang tak ternilai."

- Albert Einstein

KATA PENGANTAR

Bismillahirrahmanirrahim,

Assalamu'alaikum Warahmatullahi Wabarakatuh.

Dengan rasa syukur yang mendalam, penulis ingin mengucapkan terima kasih kepada Allah SWT dan berbagai pihak yang telah mendukung penyelesaian laporan tugas akhir ini yang berjudul " PENGEMBANGAN MODEL DETEKSI UNTUK ON-SHELF AVAILABILITY PRODUK MENGGUNGAKAN YOLOV8 PADA APLIKASI BERGERAK." Tujuan penulisan tugas akhir ini adalah untuk memenuhi persyaratan sidang tugas akhir di Jurusan Informatika, Fakultas Teknologi Industri Universitas Islam Indonesia. Penulis menyadari bahwa tanpa dukungan dan motivasi dari banyak pihak, laporan ini tidak akan terselesaikan dengan baik. Oleh karena itu, penulis ingin mengucapkan terima kasih kepada:

1. Allah SWT yang memberikan rahmat, petunjuk, dan kekuatan dalam menyelesaikan laporan ini.
2. Orang tua, Bapak Bimo Nugroho dan Ibu Sri Hardjanti, atas dukungan materiil dan emosional yang tak terhingga.
3. Saudara saya Mikael Hanugrah Brahmantya yang telah membantu saya menjaga kedua orang tua saya ketika saya menyelesaikan tugas akhir ini.
4. Bapak Prof. Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia.
5. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., selaku Ketua Jurusan Informatika Universitas Islam Indonesia.
6. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., selaku Ketua Program Studi Informatika Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia sekaligus dosen pembimbing yang telah banyak membantu dalam proses penelitian hingga penulisan tugas akhir ini.
7. Seluruh dosen Program Studi Informatika yang telah memberikan ilmu yang bermanfaat selama peneliti menuntut ilmu di UII.
8. Saudari Kartika Salma yang telah menemani dan membantu saya selama menyelesaikan tugas akhir ini.

9. Semua pihak yang tidak dapat penulis sebutkan satu per satu, terima kasih atas segala dukungan, baik secara langsung maupun tidak langsung dalam menyelesaikan penelitian ini.

Semoga ilmu, dukungan, bimbingan, dan doa yang telah diberikan kepada penulis mendapatkan balasan terbaik. Penulis menyadari bahwa laporan Tugas Akhir ini masih jauh dari sempurna dan memiliki banyak kekurangan. Oleh karena itu, penulis sangat terbuka akan saran dan masukkan. Harapannya, laporan Tugas Akhir ini dapat memberikan manfaat bagi semua pihak.

Yogyakarta, 23 Juli 2024

(Gabriel Imam Andaru)

SARI

Pengelolaan ketersediaan produk di rak merupakan tantangan utama bagi toko retail dalam memastikan kepuasan pelanggan dan meningkatkan efisiensi operasional. Kehilangan penjualan akibat kekosongan stok dapat berdampak signifikan pada pendapatan dan reputasi toko. Oleh karena itu, diperlukan sebuah sistem yang efektif dan efisien untuk memantau ketersediaan produk secara real-time. Penelitian ini bertujuan untuk mengembangkan sistem deteksi ketersediaan produk di rak (On-Shelf Availability) menggunakan teknologi YOLOv8 pada aplikasi bergerak. Sistem ini dirancang untuk membantu toko retail dalam mengoptimalkan manajemen inventaris dan mengurangi kejadian kekosongan stok.

Dalam pengembangan sistem, digunakan perangkat bergerak Samsung A6 dan Samsung A54 untuk melakukan perbandingan kinerja. Model YOLOv8 dilatih menggunakan dataset gambar produk retail yang bervariasi, kemudian diintegrasikan ke dalam aplikasi bergerak berbasis Android. Sistem ini mampu mendeteksi dan mengidentifikasi produk yang tersedia maupun yang kosong di rak berdasarkan input gambar dari kamera perangkat bergerak. Model YOLOv8 yang digunakan adalah YOLOv8n dan YOLOv8s. Hal ini didasari dengan pertimbangan bahwa kedua model tersebut memiliki waktu inferensi yang relatif lebih rendah daripada model YOLOv8 lainnya.

Hasil penelitian menunjukkan bahwa YOLOv8s memiliki hasil yang lebih bagus daripada YOLOv8n dengan masing-masing mAP50 sebesar 0,962 dan 0,947. Selain itu, penelitian ini juga menunjukkan perbedaan kinerja antara kedua perangkat. Samsung A54 menunjukkan performa yang lebih unggul dengan rata-rata waktu proses deteksi 0,075 detik. Sementara itu, Samsung A6 memiliki rata-rata waktu proses 0,341 detik.

Kesimpulan dari penelitian ini adalah bahwa sistem deteksi On-Shelf Availability produk menggunakan YOLOv8 pada aplikasi bergerak memiliki kinerja yang memuaskan dan layak diterapkan dalam pengelolaan inventaris di toko retail. Implementasi sistem ini diharapkan dapat memberikan manfaat signifikan dalam operasional toko, mengurangi kekosongan stok, dan meningkatkan pengalaman belanja konsumen.

Kata kunci: Deteksi ketersediaan produk, YOLOv8, aplikasi bergerak, toko retail, manajemen inventaris.

GLOSARIUM

CPU	perangkat keras pada komputer yang bertujuan untuk menerima dan melaksanakan perintah dari perangkat lunak komputer.
GPU	perangkat keras pada komputer yang bertujuan untuk melakukan proses komputasi dengan kecepatan tinggi.
Hyperparameter	variabel yang digunakan untuk melakukan konfigurasi saat pelatihan.
Epoch	satu putaran penuh dataset saat melakukan pelatihan model.
Loss	sebuah nilai yang mengukur tingkat kesalahan prediksi model dengan label sebenarnya
Batch	jumlah sampel dalam sebuah dataset

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Operasi ritel menjadi area penelitian yang signifikan dalam beberapa tahun terakhir. Terdapat beberapa faktor pendorong dari kemunculan ini. Ritel merupakan sektor ekonomi yang penting di mana perubahan pasar, perusahaan, proses, dan produk terjadi dengan cepat (Hübner et al., 2018). Operasi ritel teresbut dapat memberikan akses kepada pelanggan untuk mendapatkan barang belanjaan yang mereka inginkan di lokasi rak yang sesuai ketika mereka mengunjungi toko (Fisher et al., 2000). Oleh karena itu, manajemen ritel menjadi suatu tantangan sendiri karena banyaknya hal yang harus dikerjakan dalam suatu waktu seperti manajemen ketersediaan dan penataan produk di dalam rak. Ketesediaan produk tersebut biasa disebut dengan OSA (*On Shelf Availability*) yang mana mengukur kemungkinan ketersediaan sebuah produk yang dapat dijual ketika seorang pelanggan mencarinya di rak (Chopra & Meindl, 2002).

Selain itu, penataan produk di rak juga memiliki beberapa aturan yang telah ditentukan sebelumnya oleh pemilik perusahaan ritel. Penataan produk tersebut dinamakan sebagai planogram. Planogram berfungsi untuk menggambarkan tata letak penempatan setiap produk di rak dan menunjukkan berapa banyak "facings" yang seharusnya ada, yaitu berapa banyak unit stok (SKU) dari produk yang sama yang seharusnya terlihat di baris depan rak (Tonioni & Di Stefano, 2017). Penataan produk ini diatur sedemikian rupa sehingga tidak ada kekosongan stok dan memaksimalkan penjualan. Kekosongan stok (*Out-of-Stock* atau OOS) terjadi ketika seorang pelanggan di toko ritel ingin membeli produk yang tidak tersedia di dalam rak (Spielmaker et al., 2012). Hal ini menyebabkan seorang pelanggan pergi meninggalkan toko ritel tersebut dan berpindah ke toko ritel lain.

Sayangnya, untuk memastikan ketersediaan barang yang tinggi di rak, para pekerja toko harus bergerak di sekitar toko dan mencari produk yang perlu diisi ulang dan produk yang tertempatkan dengan tidak benar (Afram, 2020). Namun, tugas ini cukup memakan waktu karena biasanya ada ratusan produk yang berbeda yang harus dipantau di dalam toko.

Saat ini, pengenalan kode batang (Sriram et al., 1996) merupakan teknologi yang paling banyak digunakan, tidak hanya dalam penelitian, tetapi juga dalam industri di mana identifikasi otomatis komoditas digunakan. Dengan memindai tanda kode batang pada setiap kemasan

produk, pengelolaan produk dapat dengan mudah difasilitasi. Biasanya, hampir setiap barang di pasaran memiliki kode batang yang sesuai. Namun, karena ketidakpastian posisi pencetakan kode batang, seringkali diperlukan waktu cukup lama untuk mengidentifikasi kode batang secara manual.

Menerapkan pengenalan produk otomatis melalui gambar di toko-toko bahan makanan memiliki dampak signifikan pada industri ritel terutama dari sudut pandang motorist atau orang yang melakukan pengecekan produk karena mereka akan lebih banyak menggunakan teknologi ini untuk mengecek ketersediaan produk perusahaannya. Hal ini tentunya juga akan membantu kecocokan planogram produk di rak dengan memastikan bahwa produk ditempatkan sesuai dengan strategi pemasaran dan penjualan, serta mematuhi perjanjian dengan pemasok. Misalnya, mengenali produk dengan menggunakan deteksi produk dapat mengidentifikasi barang-barang yang hilang dari rak untuk mengingatkan staf toko untuk segera mengisi ulang produk tersebut. Bagi motorist, kemampuan ini sangat penting untuk memastikan bahwa produk mereka selalu tersedia dan ditempatkan dengan benar, sehingga memaksimalkan penjualan. Meskipun begitu, tugas mengenali produk ritel di rak masih dianggap sebagai tantangan dari sudut pandang visi komputer. Contohnya adalah jumlah kategori produk bisa sangat besar di supermarket. Menurut Goldman dkk. (Goldman et al., 2019), sebuah supermarket biasanya bisa memiliki lebih dari ribuan produk yang berbeda. Selain itu, produk-produk yang berbeda dapat memiliki penampilan yang mirip.

Deteksi objek merupakan salah satu tugas pokok dalam bidang visi komputer yang bertujuan untuk menentukan lokasi objek-objek tertentu. Hal ini sangat berguna dalam konteks ritel, seperti mengenali produk di rak untuk memberikan informasi ulasan atau harga, dan dalam navigasi di supermarket untuk meningkatkan penjualan. Sebuah studi (K. Li et al., 2021) mengulas tentang penggunaan deteksi objek dalam memantau jumlah produk di rak, mengisi produk yang hilang, dan menjalankan pencocokan dengan planogram secara berkelanjutan dengan menggunakan YOLOv2. Studi lain (Sinha et al., 2022) menggabungkan pendekatan Faster R-CNN untuk deteksi objek dan Resnet-18 untuk klasifikasi objek, dengan memanfaatkan *dataset* GroZi 120k (George & Floerkemeier, 2014) dan GP-180 (Sinha et al., 2022; Tonioni & Di Stefano, 2017).

Kecerdasan Buatan (AI) telah mengalami perkembangan pesat dalam beberapa tahun terakhir, berkat potensi besar dalam berbagai aplikasi dan terdapat tren baru dalam menjalankan pembelajaran mesin pada sistem ringan dan tertanam seperti ponsel cerdas demi mobilitas tinggi, biaya rendah, implementasi cepat, dan manfaat lainnya. Kemampuan jaringan

saraf konvolusional (CNN) berbasis ponsel cerdas menjadi teknologi yang memungkinkan untuk berbagai kasus penggunaan baru yang didukung oleh pembelajaran mesin, di mana pengenalan objek diperlukan di area luar ruangan, di mana ada faktor pembatas lain seperti kebutuhan kebebasan bergerak bagi pengguna atau infrastruktur terbatas dan cakupan yang tersedia di wilayah geografis. Selain itu, karena perangkat bergerak memiliki baterai, CPU, dan memori yang terbatas, sangat penting untuk mengoptimalkan pengunaan sumber daya agar tetap menjaga kualitas layanan yang memadai (Gunawardena et al., 2022).

Dengan munculnya minat pengembangan pembelajaran mesin dalam sistem operasi yang digunakan pada perangkat bergerak, beberapa kerangka kerja *deep learning* populer dipindahkan ke sistem operasi ini, termasuk TensorFlow Mobile (TFM), TensorFlow Lite (TFL), OpenCV, dan Qualcomm Snapdragon. Platform-platform ini ditujukan untuk menjalankan tugas inferensi di perangkat bergerak, cocok untuk skenario di mana konektivitas internet memiliki stabilitas buruk atau tidak ada (Martinez-Alpiste et al., 2022).

Deteksi objek dapat dilakukan dengan memanfaatkan berbagai algoritma, salah satunya adalah algoritma YOLOv8 (*You Only Look Once Version* 8). Kelebihan dari YOLOv8 di antaranya adalah kecepatan deteksi yang tinggi dan akurasi yang lebih baik dibandingkan dengan versi sebelumnya. Algoritma ini juga lebih efisien dalam penggunaan sumber daya komputasi, sehingga memungkinkan implementasi pada perangkat dengan kemampuan *hardware* yang terbatas. Selain itu, YOLOv8 memiliki kemampuan generalisasi yang sangat baik, yang berarti dapat mengenali objek dalam berbagai kondisi pencahayaan dan latar belakang serta dapat mengenali objek yang berukuran kecil yang mana merupakan tantangan pada munculnya YOLOv7 (Varghese & M, 2024).

Oleh karena itu, penulis mengembangkan sebuah sistem deteksi produk yang menggunakan algoritma YOLOv8 dan dapat berjalan pada aplikasi bergerak. Sistem ini dirancang untuk mengenali dan mengidentifikasi berbagai produk dengan cepat dan akurat dalam berbagai kondisi saat proses pengambilan foto yang mungkin ditemui oleh pengguna aplikasi. Dengan mengembangkan sistem ini, penulis bertujuan untuk meningkatkan efisiensi deteksi produk pada perangkat seluler, sehingga pengguna dapat dengan mudah dan cepat melakukan pengecekan ketersediaan produk hanya dengan menggunakan ponsel mereka. Selain itu, penulis berharap untuk meningkatkan akurasi deteksi produk melalui kemampuan presisi tinggi dari algoritma YOLOv8, yang diharapkan dapat meminimalisir kesalahan dalam proses identifikasi produk. Dengan demikian, pengembangan sistem deteksi produk menggunakan algoritma YOLOv8 ini diharapkan dapat memberikan kontribusi signifikan

dalam mempermudah dan mempercepat proses identifikasi produk di perangkat seluler serta membantu melakukan pengecekan ketersediaan produk secara real-time.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, muncul beberapa pertanyaan utama yang perlu dijawab, seperti bagaimana mengembangkan model yang dapat mendeteksi ketersediaan produk di rak menggunakan algoritma YOLOv8 dalam perangkat bergerak. Selain itu, juga bagaimana memastikan akurasi dan keandalan model deteksi ini menggunakan algoritma YOLOv8 pada perangkat bergerak, serta cara mengukur tingkat keberhasilannya.

1.3 Tujuan Penelitian

Tujuan Penelitian dapat dilihat sebagai berikut:

a. Mengintegrasikan algoritma YOLOv8 ke dalam aplikasi bergerak untuk mendeteksi ketersediaan produk di rak secara *real-time*.
b. Memastikan akurasi dan keandalan model deteksi menggunakan algoritma YOLOv8 pada perangkat bergerak serta mengukur tingkat keberhasilannya.

1.4 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini dapat dilihat sebagai berikut:

a. Sistem deteksi ketersediaan produk di rak dapat membantu toko ritel dalam memantau stok produk secara *real-time*, meminimalkan kekurangan stok, dan mengoptimalkan manajemen persediaan.
b. Bagi pemilik merek produk ritel, sistem ini dapat mengurangi kerumitan dalam memantau rak-rak toko dan memberikan pemberitahuan saat produk habis, sehingga mereka dapat lebih fokus pada pelayanan pelanggan.
c. Data yang dikumpulkan melalui sistem ini dapat memberikan wawasan berharga yang dapat digunakan dalam pengambilan keputusan bisnis, termasuk perencanaan persediaan, tata letak toko, dan peningkatan efisiensi operasional.

1.5 Batasan Masalah

Guna memfokuskan isu yang dibahas, penelitian ini memiliki beberapa batasan:

a. Penelitian ini akan menggunakan algoritma YOLOv8 sebagai teknologi inti untuk deteksi ketersediaan produk secara *real-time*.

b. Pengujian performa deteksi menggunakan metrik mean Average Precision (mAP) dan waktu inferensi dalam milisekon untuk mengukur kinerja dalam mendeteksi produk yang berbeda-beda.
c. Pengujian akan dilakukan menggunakan perangkat GPU NVIDIA Tesla V100, Samsung A54, dan Samsung A6.

1.6 Metodologi Secara Umum

a. *Literature review*/kajian pustaka, tahap ini melibatkan identifikasi referensi-referensi yang relevan dan membandingkannya dengan penelitian yang akan dilakukan.
b. Pengumpulan data, yang nantinya akan digunakan untuk melakukan pelatihan model dan evaluasi model dilakukan dengan cara mengunjungi beberapa toko ritel dan melakukan pengambilan beberapa foto produk pada rak toko ritel tersebut.
c. Pelabelan data, data yang telah dikumpulkan sebelumnya akan dilabeli menggunakan LabelImg.
d. Pemilihan *pre-trained* model, model yang akan digunakan adalah model YOLOv8 dengan ukuran yang berbeda beda, tetapi dengan *hyperparameter* yang sama.
e. Evaluasi, model yang telah dibangun sebelumnya akan dievaluasi menggunakan data evaluasi dan menggunakan komputasi pada CPU, GPU dan perangkat bergerak.

1.7 Sistematika Penulisan

Bagian ini berisi panduan mengenai urutan dan sistematika penulisan yang digunakan. Berikut adalah sistematika penulisan yang diterapkan:

BAB I PENDAHULUAN

Berisi latar belakang masalah yang menjadi dasar penelitian pentingnya deteksi produk pada toko ritel dan penggunaan deteksi objek pada perangkat bergerak. Berdasarakn latar belakang yang ada, kemudian dibuat rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

BAB II LANDASAN TEORI

Berisi teori-teori dan konsep yang berkaitan dengan penelitian mengenai *On Shelf Availability*, deteksi objek, CNN, fungsi aktivasi, fungsi *loss*, algoritma YOLO, dan YOLOv8. Teori-teori dan konsep bersumber dari jurnal, buku, dan artikel.

BAB III METODOLOGI PENELITIAN

Memuat tahapan dan kebutuhan penelitian yang mencakup pengumpulan data serta perancangan sistem dengan menggunakan diagram alir.

BAB IV HASIL DAN PEMBAHASAN

Memuat hasil penelitian dan analisis setiap proses yang ada, serta pengujian kinerja sistem.

BAB V KESIMPULAN DAN SARAN

Berisi kesimpulan dari hasil penelitian dan saran yang dapat diterapkan untuk penelitian selanjutnya.

BAB II
LANDASAN TEORI

2.1 OSA (*On-Shelf Availability*)

On-Shelf Availability (OSA) telah dianggap sebagai ukuran penting keberhasilan operasi bisnis ritel karena dampaknya pada permintaan saat ini dan masa depan (Gruen, 2007). Toko ritel harus memastikan tingkat *On-Shelf Availability* (OSA) yang tinggi untuk mempertahankan pelanggan jangka panjang mereka. *Out Of Stock* (OOS) dapat memiliki dampak signifikan pada keuntungan bisnis dan telah menjadi ukuran terpadu dari kinerja pengecer mengingat tenaganya, prosesnya, dan teknologinya (Jha et al., 2022). *Out Of Stock* (OOS) di ritel dapat diklasifikasikan menjadi dua subkategori. Pertama, OOS di toko, di mana barang saat ini tidak tersedia, kedua OOS di rak, di mana barang ada di toko, tetapi pelanggan tidak dapat menemukannya karena barang tidak ditempatkan di lokasi yang benar. Ketika kondisi OOS di toko terjadi, reaksi pelanggan bervariasi mulai dari beralih ke toko lain, beralih merek, bahkan tidak membeli apa-apa, yang mengakibatkan kerugian besar dalam penjualan dan pendapatan (Corsten & Gruen, 2003).

Pentingnya mengelola OSA dengan efektif telah didokumentasikan dalam berbagai studi. Menurut penelitian yang dilakukan oleh Campo, Gijsbrechts, dan Nisol (2003), penanganan yang tepat terhadap produk yang kosong dapat meningkatkan loyalitas pelanggan dan memperbaiki persepsi terhadap merek (Campo et al., 2003). Lebih lanjut, penelitian oleh Ton dan Raman (2010) menunjukkan bahwa integrasi teknologi informasi yang lebih baik dalam sistem inventarisasi dapat mengurangi kejadian OOS hingga 50% (Wang et al., 2010). Penelitian lain oleh Aastrup dan Kotzab (2010) menunjukkan bahwa strategi pengelolaan inventaris yang efektif dapat meminimalisir kerugian akibat OOS. Mereka menekankan pentingnya kerja sama antara pengecer dan pemasok dalam mengelola rantai pasokan untuk memastikan ketersediaan produk di rak (Aastrup & Kotzab, 2010).

Selain itu, penelitian oleh sebelumnya menyebutkan bahwa pelatihan karyawan dalam pengelolaan stok dan pemahaman tentang perilaku konsumen dapat meningkatkan tingkat OSA (Zinn & Liu, 2008). Secara keseluruhan, memastikan tingkat OSA yang tinggi tidak hanya berhubungan dengan aspek operasional toko tetapi juga strategi bisnis yang lebih luas. Penerapan teknologi yang canggih, pelatihan karyawan yang tepat, dan kerja sama yang erat

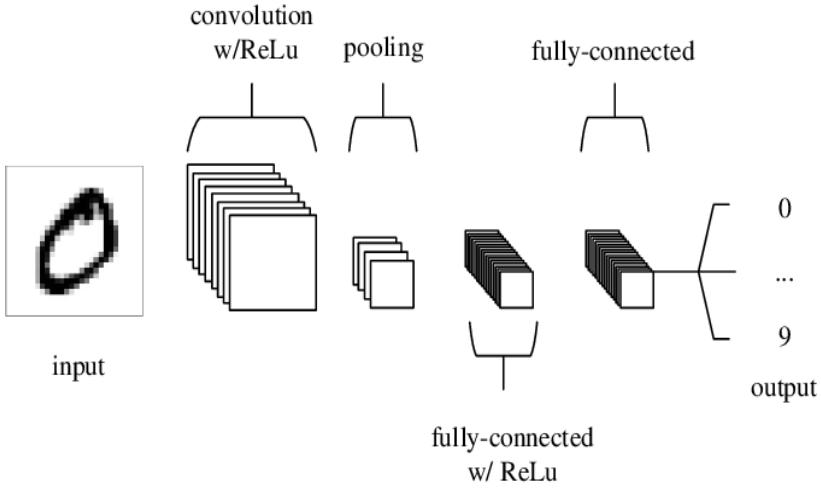
dengan pemasok merupakan beberapa pendekatan yang dapat membantu pengecer mempertahankan tingkat OSA yang optimal.

2.2 Object Detection

Object Detection, atau yang dikenal dalam bahasa Indonesia sebagai pengenalan objek, adalah sistem yang mampu mengidentifikasi objek yang terdapat dalam gambar atau video. Pengenalan objek memiliki berbagai aplikasi dalam bidang visi komputer, termasuk dalam identifikasi wajah, pelacakan pergerakan orang dalam video, atau pelacakan objek lainnya. Pada dasarnya, dalam bidang deteksi objek, sistem melakukan pencarian pada setiap bagian gambar untuk menemukan lokasi yang mirip dengan data pelatihan. Ketika gambar yang dianalisis memiliki kemiripan dengan data pelatihan, sistem dapat menghasilkan output yang mencantumkan nama objek yang berhasil dikenali. Tingkat kemiripan antara gambar input dan data pelatihan dapat diukur melalui korelasi. Deteksi objek visual melibatkan dua tugas utama: klasifikasi gambar dan lokalisasi. Berbeda dengan klasifikasi gambar sederhana atau klasifikasi yang disertai dengan lokalisasi, tugas ini lebih rumit karena biasanya terdapat beberapa objek dari berbagai kategori dalam satu gambar. Proses ini melibatkan menemukan lokasi setiap objek dalam gambar dan memberikan label kelas yang sesuai untuk setiap objek dari daftar kelas yang telah ditentukan (Amjoud & Amrouch, 2023).

2.3 *Convolutional Neural Network* (CNN)

Convolutional Neural Network atau jaringan saraf konvolusi mirip dengan *Artificial Neural Network* (ANN) tradisional dalam hal terdiri dari neuron atau sebuah fungsi matematis yang mengoptimalkan diri mereka sendiri melalui pembelajaran. Setiap neuron masih akan menerima masukan dan melakukan operasi seperti perkalian skalar diikuti oleh fungsi non linear yang mana merupakan dasar dari banyak ANN. Dari vektor gambar mentah masukan hingga output akhir berupa skor kelas, seluruh jaringan masih akan menggambarkan fungsi skor perseptual tunggal (bobot). Lapisan terakhir akan berisi fungsi-fungsi kerugian yang terkait dengan kelas-kelas, dan semua tips dan trik reguler yang dikembangkan untuk ANN tradisional masih berlaku (O'Shea & Nash, 2015). Arsitektur CNN yang disederhanakan untuk klasifikasi MNIST diilustrasikan dalam Gambar 2.1.



Gambar 2.1 Arsitektur CNN yang disederhanakan

Sumber: (O'Shea & Nash, 2015)

2.4 Fungsi Aktivasi

Fungsi aktivasi atau *activation function* adalah komponen penting dalam jaringan saraf tiruan (*neural networks*) yang digunakan untuk mengintegrasikan sinyal masukan dan menghasilkan keluaran dari neuron atau unit dalam jaringan. Fungsi aktivasi yang populer dan umum digunakan adalah Logistic Sigmoid, Tanh, ReLU, ELU, Swish, dan Mish (Dubey et al., 2022). Fungsi aktivasi memainkan peran kunci dalam CNN untuk memanfaatkan nonlinearitas guna menjaga keseimbangan antara biaya komputasi dan akurasi (Doherty et al., 2022). Sebagai contoh, fungsi aktivasi Softmax yang memiliki bentuk fungsi seperti persamaan (2.1). Fungsi ini memiliki peran untuk menghasilkan distribusi probabilitas dari kelas-kelas yang ada. Untuk setiap kelas, fungsi ini akan menghasilkan probabilitas dengan angka diantara 0 hingga 1 dengan cara melakukan eksponensial pada vektor input pada setiap kelas dan dibagi dengan jumlah eksponensial pada vektor untuk semua kelas.

$$f_i(z) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \tag{2.1}$$

Keterangan:

$f_i(z)$: fungsi softmax untuk input vektor ke- i

e^{z_i} : eksponen input vektor ke- i

$\sum_{j=1}^{n} e^{z_j}$: jumlah dari eksponen untuk seluruh elemen di vektor z

2.5 Fungsi *Loss*

Fungsi *loss* atau *Loss Function* adalah sebuah fungsi yang mengukur perbedaan antara prediksi model dengan nilai target yang sebenarnya. Tujuan dari *loss* function adalah untuk mengevaluasi seberapa baik model dalam melakukan prediksi. Idealnya, nilai *loss* function harus sekecil mungkin, menunjukkan bahwa prediksi model mendekati nilai target yang sebenarnya. Contoh dari fungsi ini adalah *Binary Cross-Entrophy Loss* yang meruakan fungsi kerugian yang umum digunakan dalam tugas klasifikasi biner di bidang pembelajaran mesin dan jaringan saraf tiruan. Fungsi ini mengukur seberapa baik model klasifikasi biner memprediksi probabilitas kelas yang benar dibandingkan dengan label target yang sebenarnya. Secara matematis, fungsi tersebut dapat ditulis seperti pada persamaan (2.2).

$$BCE\ (y', y) = -[y\ (\log(y') + (1 - y)\log(1 - y')] \tag{2.2}$$

Keterangan:

$BCE\ (y', y)$: fungsi *Binary Cross-Entrophy* untuk prediksi dan nilai target
y' : label atau nilai target
y : hasil prediksi model

Fungsi ini akan memberikan perbedaan probabilitas label target *y'* dengan label yang sebenarnya y yang bernilai 0 atau 1. Dalam objek deteksi, fungsi *loss* ini biasa digunkan untuk membedakan jenis objek yang telah terdeteksi sebelumnya.

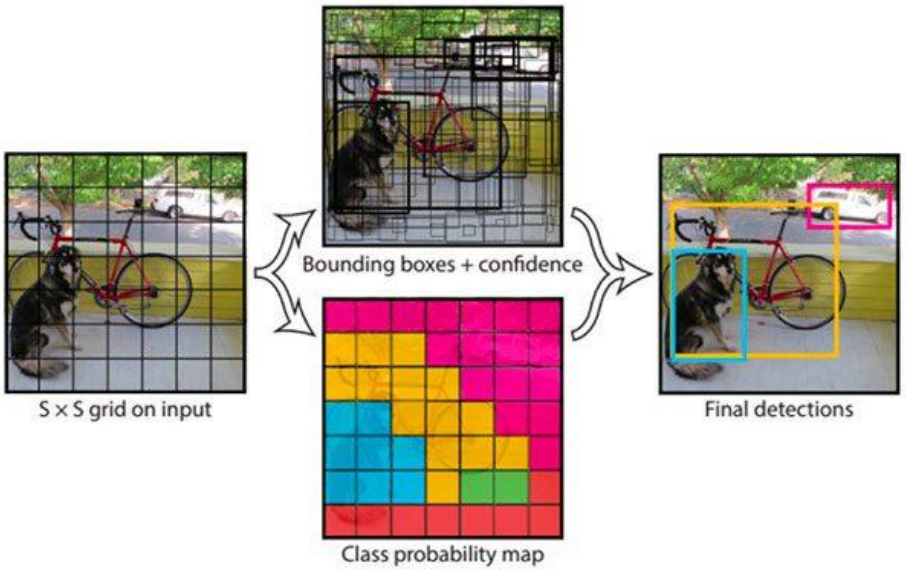
2.6 YOLO

YOLO (*You Only Look Once*), merupakan sebuah pendekatan baru dalam deteksi objek. Sebelumnya, penelitian terdahulu dalam deteksi objek memanfaatkan ulang klasifikasi untuk melakukan deteksi. Sebaliknya, pendekatan ini mengkonseptualisasikan deteksi objek sebagai masalah regresi untuk memisahkan secara spasial *bounding box* dan probabilitas kelas terkait. Sebuah jaringan saraf tunggal memprediksi kotak pembatas dan probabilitas kelas secara langsung dari gambar lengkap dalam satu evaluasi. Karena seluruh alur deteksi merupakan bagian dari satu jaringan tunggal, maka dapat dioptimalkan dari awal hingga akhir secara langsung pada kinerja deteksi (Redmon et al., 2016). Dalam pendekatan YOLO, komponen-komponen terpisah dalam deteksi objek digabungkan menjadi satu jaringan saraf tunggal. Jaringan ini memanfaatkan fitur-fitur dari seluruh gambar untuk memprediksi setiap kotak

pembatas. Selain itu, jaringan ini juga memprediksi seluruh kotak pembatas untuk semua kelas dalam satu gambar secara bersamaan. Ini berarti jaringan tersebut melakukan pemikiran secara global terhadap seluruh gambar dan semua objek dalam gambar tersebut.

YOLO (You Only Look Once), merupakan sebuah pendekatan baru dalam deteksi objek. Sebelumnya, penelitian terdahulu dalam deteksi objek memanfaatkan ulang klasifikasi untuk melakukan deteksi. Sebaliknya, pendekatan ini mengkonseptualisasikan deteksi objek sebagai masalah regresi untuk memisahkan secara spasial *bounding box* dan probabilitas kelas terkait. Sebuah jaringan saraf tunggal memprediksi kotak pembatas dan probabilitas kelas secara langsung dari gambar lengkap dalam satu evaluasi. Karena seluruh alur deteksi merupakan bagian dari satu jaringan tunggal, maka dapat dioptimalkan dari awal hingga akhir secara langsung pada kinerja deteksi (Redmon et al., 2016). Dalam pendekatan YOLO, komponen-komponen terpisah dalam deteksi objek digabungkan menjadi satu jaringan saraf tunggal. Jaringan ini memanfaatkan fitur-fitur dari seluruh gambar untuk memprediksi setiap kotak pembatas. Selain itu, jaringan ini juga memprediksi seluruh kotak pembatas untuk semua kelas dalam satu gambar secara bersamaan. Ini berarti jaringan tersebut melakukan pemikiran secara global terhadap seluruh gambar dan semua objek dalam gambar tersebut.

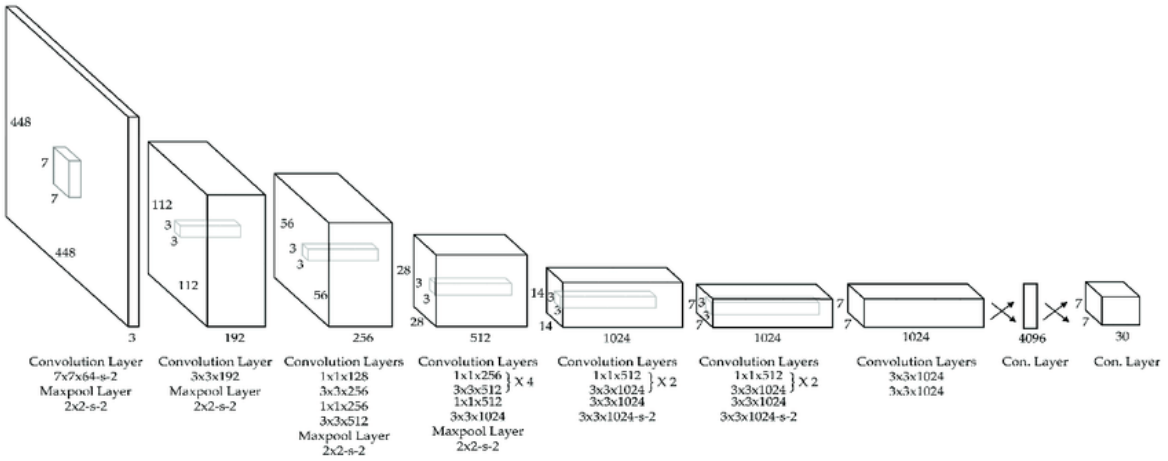
Desain YOLO memungkinkan pelatihan dari awal hingga akhir dan mencapai kecepatan waktu nyata sambil menjaga akurasi yang tinggi. Sistem ini membagi gambar masukan menjadi *grid* berukuran S × S. Apabila pusat objek berada dalam sel *grid*, maka sel *grid* tersebut bertugas untuk mendeteksi objek tersebut. Setiap sel *grid* memprediksi B kotak pembatas dan skor kepercayaan untuk kotak-kotak tersebut. Skor kepercayaan ini mencerminkan tingkat keyakinan model bahwa kotak tersebut berisi objek dan seberapa akurat kotak tersebut diprediksi (Redmon et al., 2016). Gambar 2.2 menunjukkan bahwa YOLO memperlakukan masalah deteksi objek seperti regresi.



Gambar 2.2 Proses Deteksi YOLO

Sumber: (Redmon et al., 2016)

YOLO membagi gambar menjadi *grid* berukuran S × S dan untuk setiap sel *grid* memprediksi B kotak pembatas, kepercayaan (confidence) untuk kotak-kotak tersebut, dan probabilitas kelas sebanyak C. Prediksi ini dienkoding sebagai tensor S × S × (B ∗ 5 + C). Arsitektur jaringan ini terinspirasi oleh model GoogLeNet untuk klasifikasi gambar (Szegedy et al., 2014). Jaringan kami terdiri dari 24 lapisan konvolusi yang diikuti oleh 2 lapisan terhubung penuh (*fully connected layers*). Arsitektur tersebut dapat dilihat pada Gambar 2.3.



Gambar 2.3 Arsitektur YOLO

Sumber: (Redmon et al., 2016)

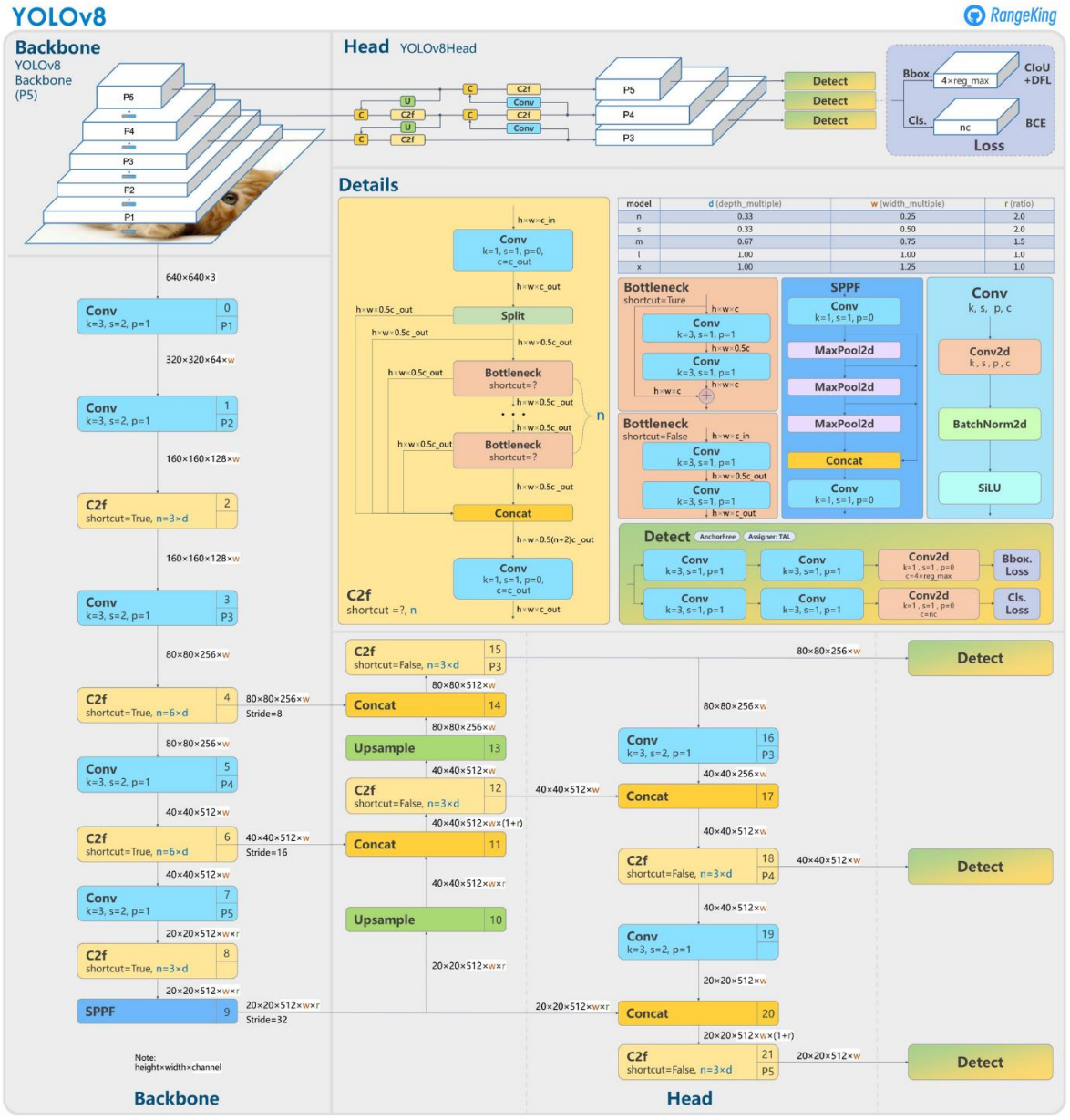
Pendekatan YOLO banyak digunakan dalam berbagai aplikasi, terutama karena kemampuan inferensi yang lebih cepat daripada mempertimbangkan akurasi deteksi. Sebagai contoh, akurasi deteksi YOLO adalah 63,4 dan Fast-RCNN 70, tetapi waktu inferensi YOLO sekitar 300 kali lebih dibanding Fast-RCNN (Diwan et al., 2023).

2.7 YOLO V8

Keluarga model YOLO (*You Only Look Once*) telah mengalami beberapa perkembangan signifikan sejak YOLOv1 pertama kali diperkenalkan pada tahun 2016. Setiap versi berusaha untuk meningkatkan akurasi dan kecepatan dalam deteksi objek, dengan peningkatan pada arsitektur jaringan, teknik pelatihan, dan pengoptimalan. YOLOv5, yang diperkenalkan pada tahun 2020 oleh Glenn Jocher dan timnya, merupakan salah satu kemajuan terbaru dalam keluarga YOLO, dengan peningkatan kinerja yang signifikan (Jocher, 2020). Model YOLOv8, yang berdasarkan arsitektur YOLOv5, memperluas kemajuan ini dengan menggabungkan tambahan inovasi dalam teknologi deteksi objek. YOLOv8 mengintegrasikan beberapa perbaikan dari model YOLOv5, termasuk waktu pelatihan yang lebih cepat, akurasi yang lebih tinggi, dan kinerja yang lebih baik pada perangkat keras yang lebih kecil (Jocher et al., 2023). Ultralytics adalah perusahaan yang mengkonversi versi-versi sebelumnya dari YOLO ke salah satu kerangka kerja terkenal dalam bidang *deep learning*, yaitu PyTorch yang ditulis dalam bahasa Python. YOLOv8 memiliki lima versi yang di-*scaling*, sesuai dengan kebutuhan aplikasi yang berbeda, yaitu YOLOv8n (*nano*), YOLOv8s (*small*), YOLOv8m (*medium*), YOLOv8l (*large*), dan YOLOv8x (*extra large*) (Jocher et al., 2023).

YOLOv8 menggunakan model tanpa *anchor* dengan kepala terpisah untuk menangani secara independen tugas deteksi objek, klasifikasi, dan regresi. Desain ini memungkinkan setiap cabang untuk fokus pada tugasnya yang spesifik, sehingga meningkatkan akurasi keseluruhan model (Wu & Dong, 2023). YOLOv8 juga mempertahankan arsitektur dasar yang mirip dengan YOLOv5, tetapi dengan penyesuaian mendalam pada CSPLayer, yang sekarang disebut modul C2f. Modul ini merupakan singkatan dari *cross-stage partial bottleneck with two convolutions*, secara efektif menggabungkan fitur-fitur tingkat tinggi dengan informasi kontekstual, yang berkontribusi pada peningkatan akurasi deteksi. Ini adalah bagian integral dari upaya berkelanjutan dalam mengembangkan teknologi deteksi objek untuk YOLOv8, yang memperluas landasan yang telah dibangun oleh YOLOv5 (Hussain, 2024). YOLOv8 menggunakan CIoU (Zheng et al., 2020) dan DFL (X. Li et al., 2020) sebagai *loss* function sedangkan untuk *classification loss* menggunkan *Binary Cross-Entropy*. Fungsi-fungsi *loss* ini

meningkatkan kinerja deteksi objek, terutama saat berurusan dengan objek-objek yang lebih kecil. Arsitektur YOLOv8 dapat dilihat pada Gambar 2.4.



Gambar 2.4 Arsitektur YOLO v8

Sumber: (Solawetz & Francesco, 2023)

Untuk evaluasi deteksi objek, cara umum untuk menentukan apakah suatu proposal objek benar adalah dengan menggunakan metode *intersection over union* (IoU). Metode ini mengambil himpunan piksel objek yang diusulkan dan himpunan piksel objek sebenarnya, dan menghitungnya dengan operasi matematika seperti pada persamaan (2.3).

$$IoU(A,B) = \frac{A \cap B}{A \cup B}; IoU(A,B)\epsilon[0,1] \quad (2.3)$$

Keterangan:

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

A : prediksi

B : label

Secara umum, ambang batas (*threshold*) IoU > 0,5 menunjukkan apakah deteksi objek berhasil atau tidak, jika melewati ambang batas, maka dianggap berhasil dan sebaliknya. Setiap kelas objek dapat dihitung akurasi rata-rata atau *average precision* sebagai menggunakan persamaan (2.4).

$$AP = \frac{TP}{TP + FP} \quad (2.4)$$

Pada persamaan (2.5), TP merupakan jumlah *instance* positif benar atau *true positive* dan FP merupakan jumlah *instance* positif dengan label sebenarnya negatif atau *false positive* untuk sebuah kelas. Untuk suatu nilai kelas sembarang, AP = 1 akan merepresentasikan tingkat deteksi yang sempurna atau *perfect detection* dan AP = 0 merupakan yang terburuk. Oleh karena itu, metrik evaluasi *mean average precision* (mAP) atas seluruh objek dalam *dataset* dapat diungkapkan dalam bentuk operasi matematika seperti persamaan (2.5).

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k \quad (2.5)$$

Dengan demikian, metrik mAP 50:95 mengindikasikan performa objek deteksi pada berbagai ambang batas yang berbeda, mulai dari 0,5 hingga 0,95, dengan peningkatan sebesar 0,05. Demikian pula, metrik mAP 50 menggambarkan performa ketika IoU > 0,5. Dalam objek deteksi, terdapat sebuah dataset yang biasa menjadi tolak ukur seberapa bagus model tersebut bekerja yaitu "COCO Object Detection Challenge," (Padilla et al., 2020). Perbandingan metrik tiap model arsitektur YOLO v8 dapat dilihat pada Tabel 2.1. Pada tabel tersebut, dapat dilihat bahwa terdapat 5 variasi pada model YOLOv8 yang memiliki jumlah parameter yang berbeda-beda. Jumlah parameter tersebut tentunya akan memengaruhi kecepatan waktu inferensi dan mAP dari hasil deteksi. Dari hasil tersebut, dapat dilihat bahwa model YOLOv8n memiliki

jumlah parameter yang paling sedikit, tetapi memiliki waktu inferensi yang paling cepat dengan mAP yang paling rendah. Sebaliknya, model YOLOv8xl memiliki jumlah parameter yang paling banyak, tetapi memiliki waktu inferensi yang paling lama dengan mAP yang paling tinggi.

Tabel 2.1 Perbandingan model YOLOv8

Sumber: (Jocher et al., 2023)

Model	Ukuran	mAP 50-90	Waktu Inferensi GPU A100 (ms)	Jumlah Parameter (M)	FLOPs (B)
YOLOv8n	640	37,3	0,99	3,2	8,7
YOLOv8s	640	44,9	1,20	11,2	28,6
YOLOv8m	640	50,2	1,83	25,9	78,9
YOLOv8l	640	52,9	2,39	43,7	165,2
YOLOv8xl	640	53,9	3,53	68,2	257,8

2.8 Kuantisasi

Dengan meningkatnya popularitas kecerdasan buatan dan *deep learning* sebagai alat serbaguna untuk memasukkan kecerdasan ke dalam perangkat elektronik, kebutuhan akan efektifitas jaringan saraf, rendah latensi, dan hemat energi telah meningkat. Saat ini, jaringan saraf dapat ditemukan dalam banyak perangkat elektronik dan layanan, mulai dari perangkat bergerak, kacamata cerdas, dan peralatan rumah tangga, hingga drone, robot, dan mobil otonom. Perangkat-perangkat ini biasanya memiliki beberapa limitasi seperti batasan waktu dalam melakukan komputasi jaringan saraf atau memiliki daya yang rendah untuk kinerja jangka panjang (Xu et al., 2018).

Salah satu cara paling berdampak untuk mengurangi waktu komputasi dan konsumsi daya saat menggunakan jaringan saraf adalah dengan menggunakan teknik kuantisasi (Cheng et al., 2018). Pada kuantisasi jaringan saraf, bobot dan tensor aktivasi disimpan dalam presisi bit yang lebih rendah dibandingkan dengan presisi 16 atau 32 bit yang biasanya digunakan selama pelatihan. Peralihan dari 32 bit ke 8 bit mampu mengurangi memori ekstra penyimpanan tensor sebesar faktor 4, sementara biaya komputasi perkalian matriks berkurang secara kuadrat sebesar faktor 16 (Nagel et al., 2021). Jaringan saraf telah terbukti tangguh terhadap kuantisasi, artinya mereka dapat dikuantisasi ke ukuran bit yang lebih rendah dengan dampak yang relatif kecil pada akurasi jaringan. Selain itu, kuantisasi pada jaringan saraf

seringkali dapat diterapkan bersamaan dengan metode umum lainnya untuk optimalisasi jaringan saraf, seperti pencarian arsitektur jaringan saraf, kompresi, dan *pruning*. Kuantisasi adalah langkah penting dalam pipa efisiensi model untuk penggunaan praktis dalam *deep learning*. Namun, kuantisasi pada jaringan saraf bukanlah proses yang tidak memiliki kekurangan. Kuantisasi dengan ukuran bit yang lebih rendah mengakibatkan *noise* atau bias ke dalam jaringan yang dapat mengakibatkan penurunan akurasi. Meskipun beberapa jaringan tangguh terhadap *noise* atau bias, jaringan lain memerlukan pekerjaan ekstra untuk mengoptimalkan manfaat dari kuantisasi.

2.9 Kajian Pustaka

Terdapat berbagai penelitian yang mendalam mengenai pengenalan produk di lingkungan ritel dengan menggunakan berbagai metode dan *dataset*. Sebagai contoh, Ankit Sinha, dkk. (2022) melakukan penelitian yang berjudul "An Improved Deep Learning Approach For Product Recognition on Racks in Retail Stores." Mereka mengembangkan pendekatan *deep learning* yang lebih baik untuk mengenali produk di rak toko ritel. Dalam penelitian ini, mereka menggabungkan metode Faster R-CNN untuk deteksi objek dan Resnet-18 untuk klasifikasi objek, serta menggunakan Grozi Dataset dan GP-180 Dataset. Hasilnya menunjukkan tingkat akurasi yang mengesankan, dengan Grozi Dataset mencapai 47.77% mAP dan GP-180 Dataset mencapai 82.70% mAP (Sinha et al., 2022).

Selain itu, Prabu Selvam, dkk. (2022) juga melakukan kontribusi penting dalam bidang ini dengan penelitian yang berjudul "A Deep Learning Framework for Grocery Product Detection and Recognition." Mereka mengembangkan sebuah kerangka kerja *deep learning* dengan menggunakan metode YOLOv5 untuk mendeteksi dan mengenali produk di toko kelontong. Penelitian ini memanfaatkan dataset Grozi 120k dan WebMarket. Hasil penelitian ini menunjukkan akurasi yang sangat baik, dengan GroZi mencapai 86.3% pada metrik presisi, 77.8% pada *recall*, dan 77.04 pada F1-*score*. Sementara Webmarket mencapai 89.4% pada presisi, 88.2% pada *recall*, dan 86.26 pada F1-s*core* (Selvam et al., 2022).

Muhammad Nashir Ardiansyah, dkk. (2021) juga memiliki kontribusi penting dalam bidang ini dengan penelitian yang berjudul "Object/Product Identification for Stock Taking Activities using Object Recognition Concept." Mereka menggunakan metode YOLOv3 untuk mengidentifikasi objek atau produk dalam konteks kegiatan pengambilan stok. Meskipun menggunakan *dataset* privat, penelitian ini berhasil mencapai rasio deteksi sebesar 81%, menunjukkan potensi besar untuk aplikasi praktis di lingkungan ritel (Ardiansyah et al., 2021).

Selain itu, Ceren Gulra Melek, dkk. (2019) fokus pada deteksi objek dalam gambar rak toko dengan penelitian berjudul "Object Detection in Shelf Images with YOLO." Mereka menggunakan metode YOLOv2 dan *dataset* Grocery. Hasilnya menunjukkan bahwa meskipun mengalami *loss* sebesar 29,84% setelah 900 iterasi, metode ini memiliki potensi untuk pengembangan lebih lanjut (Melek et al., 2019).

Terakhir, Wei Yi, dkk. (2019) mencoba mendeteksi produk di lokasi penjualan tanpa perlu pelabelan manual oleh manusia dengan penelitian berjudul "Detecting retail products in situ using CNN without human effort labeling." Mereka menggunakan metode Faster R-CNN dan *dataset* yang mencakup 324 kategori produk, masing-masing dengan 5000 gambar. Meskipun tidak mencapai tingkat akurasi tertinggi, penelitian ini menunjukkan kemungkinan penggunaan teknologi tanpa campur tangan manusia dalam pengenalan produk di lingkungan ritel. Secara keseluruhan, penelitian-penelitian ini mencerminkan perkembangan yang signifikan dalam penggunaan *deep learning* dan teknik deteksi objek untuk aplikasi di industri ritel (Yi et al., 2019). Berikut ringkasan penelitian terdahulu yang disajikan dalam Tabel 2.2.

Berdasarkan penelitian-penelitian sebelumnya, penelitian ini berfokus pada pengembangan sistem deteksi objek dengan menggunakan YOLOv8 untuk meningkatkan ketersediaan produk di rak (*on shelf availability*) melalui aplikasi bergerak, yang diharapkan dapat memberikan solusi yang lebih akurat dan efisien dibandingkan metode sebelumnya. Penggunaan model YOLOv8 ini diharapkan dapat memperbaiki hasil deteksi sebelumnya dikarenakan model YOLOv8 merupakan model yang cukup baru. Selain itu, penggunaan objek deteksi pada aplikasi bergerak dapat diharapkan menjadi sebuah metode baru yang akan digunakan pada industri ritel untuk mengatasi keterbatasan koneksi internet yang ada saat melakukan pengecekan ketersediaan produk.

Tabel 2.2 Penelitian terdahulu mengenai pengenalan produk pada toko ritel

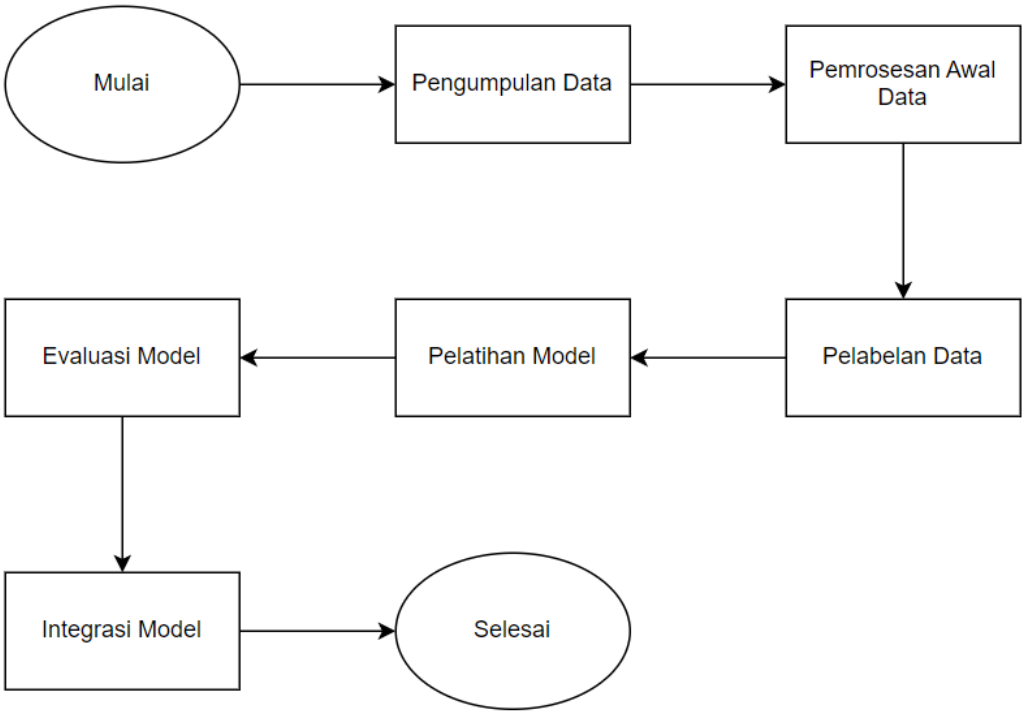
No	Peneliti	Tahun	Judul	Metode	*Dataset*	Hasil
1	Ankit Sinha, Soham Banerjee dan Pratik Chattopadhyay	2022	An Improved Deep Learning Approach for Product Recognition on	Faster R-CNN untuk Deteksi Objek dan Resnet-18 untuk	Grozi Dataset dan GP-180 Dataset	Grozi Dataset 47.77 % mAP GP-180 Dataset 82.70 % mAP

			Racks in Retail Stores	Klasifikasi Objek		
	Prabu Selvam, Joseph Abraham Sundar Koilraj	2022	A Deep Learning Framework for Grocery Product Detection and Recognition	YOLOv5	Grozi 120k Dataset, WebMarket Dataset	GroZi = presisi 86.3, *recall* 77.8, dan F1-*score* 77.04 Webmarket = presisi 89.4, *recall* 88.2, dan F1-*score* 86.26
	Muhammad Nashir Ardiansyah, Prafajar Sukssesanno Muttaqin, Murman Dwi Prasetio, Nia Novitasari	2021	Object/Product Identification for Stock Taking Activities using Object Recognition Concept	YOLOv3	Private Dataset	Rasio Deteksi mencapai 81 %
	Ceren Gulra Melek, Elena Battini Sonmez, Songul Albayrak	2019	Object Detection in Shelf Images with YOLO	YOLOv2	Grocery Dataset	Mendapatkan *loss* sebesar 29,84% dengan 900 iterasi
	Wei Yi, Yaoran Sun,	2019	Detecting retail products in situ	Faster R-CNN	324 Kategori	0,86 mAP

	Tao Ding, Sailing He		using CNN without human effort labeling		produk dengan masing masing 5000 gambar	

BAB III
METODOLOGI PENELITIAN

Penelitian ini dilakukan melalui serangkaian tahapan yang dimulai dengan pengumpulan data dari berbagai sumber. Data yang telah dikumpulkan kemudian diproses untuk meningkatkan kualitas dan kelengkapannya. Tahapan selanjutnya adalah memberikan label atau keterangan pada data untuk memudahkan proses pelatihan model. Model kemudian dilatih menggunakan data yang telah diproses dan dilabeli. Model yang telah dilatih kemudian dievaluasi untuk mengetahui kinerjanya. Terakhir, model diintegrasikan ke dalam perangkat bergerak dan diinferensikan dalam kondisi sebenarnya untuk memastikan bahwa model dapat bekerja dengan baik. Diagram alir dari tahpan penlitian tersebut dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram Alir Penelitian

3.1 Pengumpulan Data

Dataset yang digunakan adalah kumpulan data yang terdiri dari serangkaian gambar yang mencakup beberapa kelas produk yang tersusun di rak toko. *Dataset* ini terdiri dari gambar-gambar yang mewakili 125 kelas. Untuk mencapai model dengan kinerja yang akurat, penting

memiliki *dataset* yang mencerminkan kondisi gambar yang serupa dengan data uji. Oleh karena itu, kami mengambil gambar langsung di supermarket menggunakan perangkat bergerak sebagai bagian dari proses pengumpulan *dataset*.

3.1.1 Aturan Pengambilan *Dataset*

Dalam pengumpulan data ini, variasi sudut pemotretan gambar dijaga dengan cermat, termasuk sudut pandang yang umumnya tegak lurus dengan produk. Terdapat juga gambar dengan sudut pandang miring dari sisi kanan atau kiri, tetapi diupayakan tidak terlalu miring demi menjaga visibilitas produk. Beberapa gambar juga diambil dengan sudut sedikit dari atas maupun bawah produk untuk memberikan perspektif alternatif. Contoh gambar yang digunakan dalam penelitian ini dapat dilihat pada Gambar 3.2.

Selain variasi sudut pandang, pencahayaan juga menjadi faktor kunci dalam *dataset* ini. Sebagian besar gambar dalam *dataset* ini mengadopsi pencahayaan yang relatif serupa, yaitu pencahayaan dengan tingkat kecerahan yang mirip dengan kondisi pencahayaan ruangan dengan lampu yang umumnya menyala. Hal ini dapat dilihat dalam gambar-gambar yang diambil di dalam toko ritel dengan pencahayaan yang seragam. Konsistensi pencahayaan ini mendukung keseragaman dalam pengamatan visual produk, suatu aspek penting dalam menganalisis karakteristik produk yang terdapat dalam *dataset* ini.

Gambar 3.2 Contoh Gambar untuk *Dataset*

3.1.2 Kelas *Dataset*

Kelas yang digunakan dalam pengumpulan *dataset* mencakup total 125 kategori produk yang berbeda. Rincian kelas-kelas ini tersedia dalam Tabel 3.1. Setiap kelas mewakili jenis produk yang berbeda yang dikenali dan diidentifikasi oleh model deteksi objek. *Dataset* ini memungkinkan model untuk belajar dan mengklasifikasikan berbagai produk dengan akurat dalam konteks pengenalan di rak toko. Jumlah data dalam sebuah dataset tentu akan memengaruhi hasil performa model dalam melakukan deteksi. Seiring bertambahnya jumlah data, maka akan semakin bagus juga performa model dengan syarat jika data yang digunakan sesuai dengan kondisi di saat pengambilan foto yang sebenarnya (Zhu et al., 2015).

Tabel 3.1 Kelas *Dataset*

Kelas *Dataset*
NESTLE BATITA 1+ Madu+Iron 40x150g N1
NESTLE BATITA 1+ Madu+Iron 24x400g
LACTOGEN 2 Happynutri 12x750g N1 ID_REDESIGN1
LACTOGROW 4 Happynutri Honey12x750g N1ID_REDESIGN1
LACTOGROW 4 Happynutri Van 12x750g N1 ID_REDESIGN1
LACTOGEN 2 Happynutri 40x180g N1 ID_REDESIGN1
LACTOGEN 1 Happynutri 40x180g N1 ID_REDESIGN1
FRISIAN FLAG
LACTOGROW 3 Happynutri Van 24x350g N1 ID_REDESIGN1
LACTOGEN 2 Happynutri 24x350g N1 ID_REDESIGN1
LACTOGEN 1 Happynutri 12x1kg ID_REDESIGN1
LACTOGEN 2 Happynutri 12x1kg ID_REDESIGN1
LACTOGROW 3 Happynutri Honey 12x1kg N1ID_REDESIGN1
LACTOGROW 3 Happynutri Van 12x1kg ID_REDESIGN1
LACTOGROW 4 Happynutri Van 12x1kg ID_REDESIGN1
LACTOGROW 3 Happynutri Honey 12x750gN1ID_REDESIGN1
LACTOGROW 4 Happynutri Honey12x1kg ID_REDESIGN1
LACTOGROW 3 Happynutri Van 12x750g N1 ID_REDESIGN1
MORINAGA
LACTOGROW 3 Happynutri 12x750g N1 ID_REDESIGN1

LACTOGEN 1 Happynutri 12x750g N1 ID_REDESIGN1
LACTOGEN 1 Happynutri 24x350g N1 ID_REDESIGN1
LACTOGROW 3 Happynutri Honey 24x350gN1ID_REDESIGN1
DANCOW 5+ Van Advn ExcNutr 24x400g A
DANCOW 5+ Madu Advn ExcNutr 24x400g
DANCOW 5+ Cok Advn ExcNutr 24x400g
DANCOW 5+ Cok Advn ExcNutr 12x1000g
DANCOW 3+ Van Advn ExcNutr 12x800g A
DANCOW 3+ Madu Advn ExcNutr 12x800g A
DANCOW 3+ Cok Advn ExcNutr 12x800g
DANCOW 3+ Cok Advn ExcNutr 24x400g A
DANCOW 5+ Cok Advn ExcNutr 12x800g
DANCOW 5+ Madu Advn ExcNutr 12x800g
DANCOW 3+ Madu Advn ExcNutr 12x1000g
DANCOW 1+ Madu Advn ExcNutr 12x800g
DANCOW 5+ Van Advn ExcNutr 12x800g A
DANCOW 1+ Van Advn ExcNutr 12x800g N1
DANCOW 1+ Cok Advn ExcNutr 12x800g N1 A
DANCOW 1+ Van Advn ExcNutr 24x400g
DANCOW 1+ Madu Advn ExcNutr 24x400g N1
DANCOW 3+ Madu Advn ExcNutr 40x200g
NESTLE DATITA 3+ Madu+Iron 24x400g
DANCOW 3+ Van Advn ExcNutr 24x400g
DANCOW 3+ Madu Advn ExcNutr 24x400g
NESTLE BATITA 1+Vanilla+Iron 12x900g
NESTLE DATITA 3+ Madu+Iron 12x900g
LACTOGROW 3 Happynutri Honey 40x180gN1ID_REDESIGN1
LACTOGROW 3 Happynutri 24x350g N1 ID_REDESIGN1
DANCOW 1+ Cok Advn ExcNutr 24x400g N1
DANCOW 5+ Van Advn ExcNutr 24x400g ID_REDESIGN2
NESTLE BATITA 1+ Madu+Iron 12x850g ID
DANCOW 1+ Madu Advn ExcNutr 12x800g ID_REDESIGN2

NESTLE BATITA 1+ Vanilla+Iron 24x400g
NESTLE BATITA 1+ Madu+Iron 12x900g
DANCOW 3+ Madu Advn ExcNutr 12x800g ID_REDESIGN2
DANCOW 5+ Madu Advn ExcNutr 12x800g ID_REDESIGN2
DANCOW 5+ Madu Advn ExcNutr 12x1000g
NESTLE DATITA 3+ Madu+Iron 40x150g N1
NESTLE DATITA 3+ Vanilla+Iron 24x400g
SGM
VIDORAN
ENFAGROW
DANCOW 1+ Van Advn ExcNutr 12x800g N1 ID_REDESIGN2
DANCOW 3+ Van Advn ExcNutr 12x800g ID_REDESIGN2
DANCOW 3+ Madu Advn ExcNutr 12x1000g ID_REDESIGN2
DANCOW 5+ Van Advn ExcNutr 12x800g ID_REDESIGN2
DANCOW 3+ Cok Advn ExcNutr 12x800g ID_REDESIGN2
DANCOW 3+ Van Advn ExcNutr 24x400g ID_REDESIGN2
DANCOW 5+ Madu Advn ExcNutr 24x400g ID_REDESIGN2
DANCOW 3+ Van Advn ExcNutr 12x1000g ID_REDESIGN2
DANCOW 1+ Van Advn ExcNutr 12x1000g ID_REDESIGN2
DANCOW 5+ Cok Advn ExcNutr 12x800g ID_REDESIGN2
DANCOW 1+ Madu Advn ExcNutr 12x1000gN1ID_REDESIGN2
DANCOW 5+ Madu Advn ExcNutr 12x1000g ID_REDESIGN2
DANCOW 1+ Van Advn ExcNutr 24x400g ID_REDESIGN2
DANCOW 3+ Madu Advn ExcNutr 24x400g ID_REDESIGN2
DANCOW 1+ Madu Advn ExcNutr 24x400g N1ID_REDESIGN2
DANCOW 1+ Van Advn ExcNutr 40x200g N1 ID_REDESIGN2
DANCOW 3+ Madu Advn ExcNutr 40x200g ID_REDESIGN2
DANCOW 1+ Cok Advn ExcNutr 12x800g N1 ID_REDESIGN2
LACTOGROW 4 Happynutri Van 24x350g N1 ID_REDESIGN1
NESTLE DATITA 3+Vanilla+Iron 12x900g
NESTLE DATITA 5+ Madu+Iron 12x900g
DANCOW 1+ Cok Advn ExcNutr 12x800g N1

DANCOW 1+ Madu Advn ExcNutr 12x1000gN1
DANCOW 3+ Cok Advn ExcNutr 24x400g
DANCOW 3+ Van Advn ExcNutr 12x1000g
LACTOGROW 4 Happynutri Van 12x1kg ID
LACTOGROW 4 Happynutri Honey12x1kg ID
LACTOGROW 3 Happynutri Van 12x1kg ID
DANCOW 1+ Madu Advn ExcNutr 40x200g ID_REDESIGN2
LACTOGEN 1 Happynutri 40x180g N1 ID_REDESIGN2
LACTOGEN 1 Happynutri 12x750g N1 ID_REDESIGN2
LACTOGEN 1 Happynutri 24x350g N1 ID_REDESIGN2
LACTOGEN 2 Happynutri 12x750g N1 ID_REDESIGN2
LACTOGROW 3 Happynutri Honey 12x750gN1ID_REDESIGN2
LACTOGROW 4 Happynutri Honey12x750g N1ID_REDESIGN2
LACTOGROW 3 Happynutri 12x750g N1 ID_REDESIGN2
LACTOGEN 2 Happynutri 24x350g N1 ID_REDESIGN2
S-26 PROCAL GOLD Can 24x400g N8 ID
S-26 PROMIL GOLD 2 Can 24x400g N8 ID
S-26 PROMIL GOLD 1 Can 24x400g N8 ID
S-26 ULTIMA PROMIL 1 Can Top 6x850g ID
S-26 ULTIMA PROMIL 2 Can Top 6x850g ID
S-26 ULTIMA PROCAL Can Top 6x850g ID
S-26 GOLD PROMIL 1 Can Top 6x900g N8 ID
S-26 PROCAL GOLD Can Top 6x900g N8 ID
S-26 PROMISE GOLD Can Top 6x900g N8 ID
S-26 PROMISE Pouch 12x400g N4 ID_REDESIGN1
S-26 PROCAL Honey Pouch 12x400g ID
S-26 PROMIL 1 Pouch 12x700g ID
S-26 PROMIL 2 Pouch 12x700g ID
S-26 PROCAL Pouch 12x400g N4 ID
S-26 PROMIL 1 Pouch 12x400g N2 ID
S-26 PROMIL 2 Pouch 12x400g N2 ID
S-26 PROCAL Pouch 6(2x700g) N2 ID

S-26 PROMISE Pouch 12x700g N4 ID_REDESIGN1
S-26 PROCAL Pouch 12x700g N4 ID
S-26 PROCAL GOLD Can 6x1.6kg N8 ID
S-26 PROMISE GOLD Can 6x1.6kg N8 ID
S-26 GOLD PROMIL 2 Can Top 6x900g N8 ID
S-26 PROMISE Pouch 6(2x700g) N1 ID
LACTOGROW 4 Happynutri Van 12x1kg ID_REDESIGN2
LACTOGROW 3 Happynutri Van 12x750g N1 ID_REDESIGN2
DANCOW 1+ Van Advn ExcNutr 12x1000g
DANCOW 1+ Van Advn ExcNutr 40x200g N1
LACTOGROW 3 Happynutri 12x1kg N1ID_REDESIGN1
DANCOW 1+ Madu Advn ExcNutr 40x200g
S-26 PROMISE Pouch 12x400g N4 ID
S-26 PROMISE Pouch 12x700g N4 ID
NUTRICIA
DANCOW 5+ Van Advn ExcNutr 12x800g
DANCOW 3+ Madu Advn ExcNutr 12x800g
DANCOW 3+ Van Advn ExcNutr 12x800g
DANCOW 5+ Van Advn ExcNutr 24x400g

3.2 Pemrosesan Awal Data

Dalam tahap ini, peneliti memanfaatkan data gambar yang telah terkumpul, kemudian melakukan proses pengolahan awal agar gambar dapat dimasukkan dalam model. Pemrosesan ini dilakukan dengan cara mengubah ukuran gambar menjadi ukuran 1024x1024. Pemilihan ukuran gambar 1024x1024 dilakukan karena pada proses pengambilan beberapa sampel gambar, penulis menemukan bahwa rata-rata ukuran gambar yang diperoleh adalah 1024x768. Untuk melakukan simplifikasi pada proses memasukkan gambar kedalam model untuk proses pelatihan, maka dipilihlah ukuran gambar 1024x1024.

Proses pengubahan ukuran gambar tersebut dilakukan dengan menggunakan *library OpenCV* dan menggunakan fungsi *resize* yang dapat dilihat pada Gambar 3.3. Pertama, gambar dibaca dengan menggunakan fungsi *imread*. Setelah itu, gambar diubah ukurannya

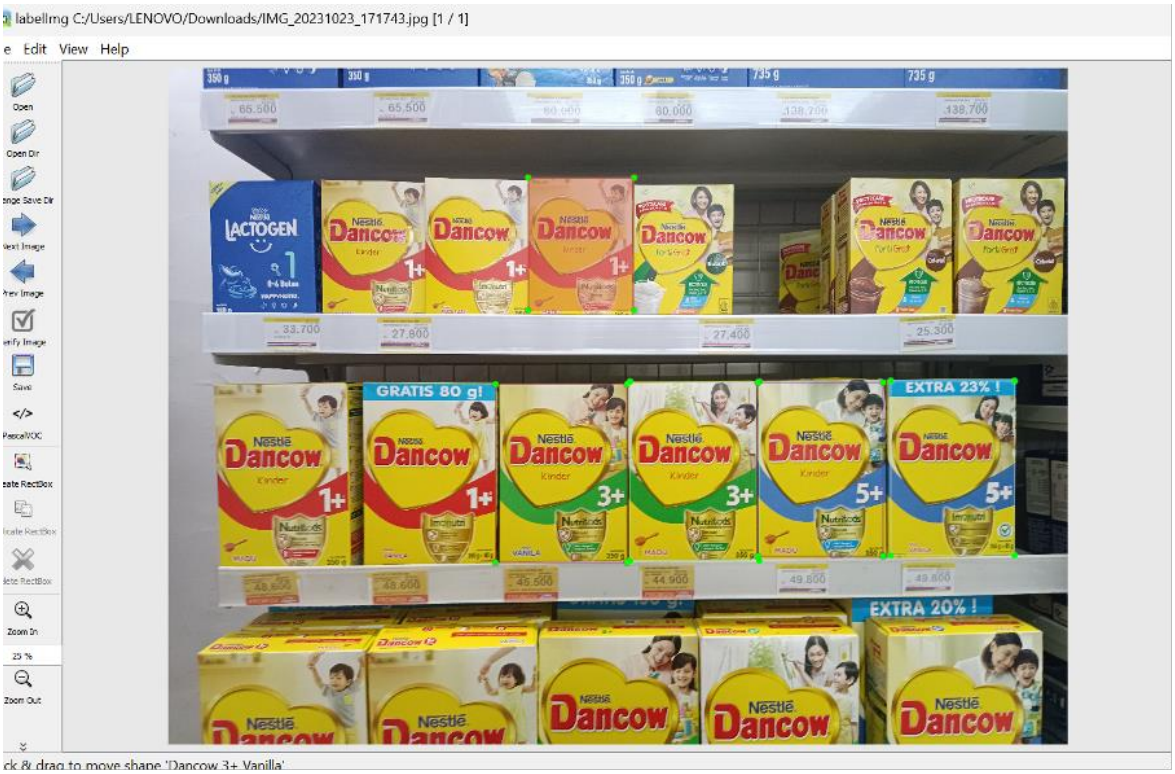
menggunakan fungsi *resize* dengan menggunakan fungsi interpolasi untuk melakukan konstruksi gambar baru dengan ukuran yang lebih besar. Setelah itu gambar disimpan menggunakan fungsi *imwrite*.

```
import cv2

def resize_image(input_path, output_path, size=(1024, 1024)):
    # Baca gambar
    img = cv2.imread(input_path)

    # Ubah ukuran gambar
    resized_img = cv2.resize(img, size, interpolation=cv2.INTER_CUBIC)

    # Simpan gambar
    cv2.imwrite(output_path, resized_img)

# Ganti nama gambar
input_image_path = 'input.jpg'
output_image_path = 'output.jpg'
resize_image(input_image_path, output_image_path)
```

Gambar 3.3 Kode Program untuk *Resize*

3.3 Pelabelan Data

Data gambar yang telah melewati tahap pemrosesan awal akan diberi label, dengan tujuan memberikan kategori atau kelas objek sebelum melanjutkan ke tahap pelatihan model. Proses pelabelan dilaksanakan secara manual pada setiap data gambar dengan menggunakan *tool* "LabelImg" yang dapat dilihat pada Gambar 3.4, dengan cara melakukan anotasi berupa kotak yang dilengkapi dengan nama kelas pada setiap objek.



Gambar 3.4 Pelabelan dengan LabelImg

Dari setiap objek, dihasilkan koordinat dua titik (x, y) yang menunjukkan sudut kiri atas (*top-left*) dan sudut kanan bawah (*bottom-right*) dari kotak tersebut. Koordinat ini memberikan informasi tentang lokasi dan ukuran objek dalam gambar. Setelah setiap gambar mendapatkan anotasi lengkap beserta nama kelas, dihasilkan sebuah direktori yang memuat seluruh gambar yang telah dianotasi dan sebuah direktori yang berisi anotasi dari gambar dengan format yang dapat dilihat pada Gambar 3.5.

<label> <x top-left> <y top-left> <x bottom-right> <y bottom-right>

Gambar 3.5 Format Anotasi YOLOv8

3.4 Pelatihan Model

Pada tahap ini, data gambar yang telah diberi label atau anotasi akan dilatih menggunakan model arsitektur YOLOv8 dan konfigurasi *hyperparameter* yang telah ditentukan. Tujuannya adalah agar model dapat memahami pola atau karakteristik dari data gambar yang telah dipersiapkan. Melalui proses ini, diharapkan dapat dihasilkan sebuah model yang mampu mendeteksi objek sesuai kelasnya dengan tingkat akurasi yang tinggi dan dapat

diimplementasikan pada perangkat bergerak. Proses pelatihan model dikodekan seperti pada Gambar 3.6.

```
/home/skripsi$ pip install ultralytics==8.0.162
/home/skripsi$ yolo detect train model=yolov8n.pt data=data.yaml imgsz=1024 batch-size=8
/home/skripsi$ yolo detect train model=yolov8s.pt data=data.yaml imgsz=1024 batch-size=8
```

Gambar 3.6 Kode Program Pelatihan Model

Penelitian ini menggunakan YOLOv8n dan YOLOv8s karena waktu inferensinya yang relatif lebih cepat daripada YOLOv8m dan yang lain seperti yang dapat dilihat pada Tabel 2.1. Dapat dilihat bahwa, selisih waktu inferensi antara YOLOv8n dengan YOLOv8s hanya terpaut 0,21 ms sedangkan selisih waktu inferensi antara YOLOv8s dengan YOLOv8m terpaut 0,63 ms. Waktu inferensi yang cepat sangat penting untuk aplikasi *real-time* karena memungkinkan sistem untuk menganalisis dan memproses data secara instan, sehingga dapat segera mendeteksi produk yang tersedia atau kosong di rak tanpa penundaan. Hal ini memastikan bahwa informasi mengenai ketersediaan produk selalu terkini dan akurat.

3.4.1 *Split Dataset*

Sebelum melakukan pelatihan model pada data gambar, perlu dilakukan pemisahan atau pembagian data menjadi dua bagian, yaitu data latih (*train*) dan data uji (*test*). Data latih digunakan untuk melatih model, sementara data uji digunakan untuk menguji kinerja model tersebut. Pembagian ini dilakukan dengan rasio populasi 90% untuk data latih dan 10% untuk data uji. Rasio ini dipilih karena jumlah gambar yang relatif sedikit dengan jumlah kelas sebanyak 125. Lalu, sejumlah 11.216 gambar akan digunakan sebagai data latih, sedangkan 615 gambar akan dijadikan sebagai data uji.

3.4.2 *Model Selection*

Dalam penelitian ini, peneliti menggunakan dua model dari YOLOv8 yaitu YOLOv8n dan YOLOv8s. Perbedaan utama antara model-model ini terletak pada ukuran model dan jumlah parameternya. YOLOv8n memiliki ukuran model terkecil sebesar 27,4 MB dengan 15,3 juta parameter, menjadikannya pilihan ideal untuk perangkat dengan sumber daya yang terbatas, seperti perangkat bergerak atau Raspberry Pi. YOLOv8s, dengan ukuran model 66,3

MB dan 21,3 juta parameter, menawarkan kinerja lebih baik dibandingkan YOLOv8n, tetapi tetap efisien untuk digunakan pada perangkat yang tidak terlalu kuat (Jocher, 2020).

3.4.3 *Hyperparameter Setting*

Dalam penelitian ini, pemilihan *hyperparameter* menjadi tahapan yang sangat penting dalam meningkatkan kinerja pelatihan model. Penelitian ini hanya berfokus pada tiga *hyperparameter* utama yaitu ukuran *batch*, jumlah *epoch*, dan dimensi gambar (*image size*). Ukuran *batch* yang dipilih adalah 32. Ukuran ini dipilih karena memungkinkan pengelolaan data secara efisien dalam setiap iterasi pelatihan. Bengio menyatakan bahwa memilih ukuran *batch* sebesar 32 adalah langkah awal yang baik. Ia juga mengemukakan bahwa ukuran *batch* yang lebih besar dapat mempercepat perhitungan jaringan, tetapi dapat mengurangi jumlah pembaruan yang diperlukan agar jaringan mencapai konvergensi (Bengio, 2012).

Sementara itu, penentuan jumlah *epoch* sebanyak 300 dirancang untuk mengidentifikasi potensi *overfitting* atau *underfitting* pada model yang dibangun. Hal ini memungkinkan untuk mengamati performa model karena *library* yang digunakan dapat menyimpan hasil dari *epoch* terbaik selama pelatihan, termasuk metrik evaluasi terbaik (Jocher, 2020). Selanjutnya, ukuran gambar yang dipilih adalah 1024x1024 piksel. Ukuran ini dipilih berdasarkan rata-rata resolusi gambar yang digunakan dan untuk meningkatkan kemampuan deteksi objek pada berbagai skala.

3.4.4 *Transfer Learning*

Proses pelatihan model YOLOv8 dilakukan dengan menerapkan teknik *transfer learning*. *Transfer learning* dalam deteksi objek melibatkan penggunaan model yang telah dilatih sebelumnya pada *dataset* besar dan menyesuaikannya kembali pada *dataset* yang lebih kecil yang spesifik untuk tugas yang sedang dilakukan. Pendekatan ini dapat menghemat waktu dan sumber daya komputasi, karena model yang telah dilatih sebelumnya sudah mempelajari untuk mengekstrak fitur dari gambar. Melalui penyesuaian model pada *dataset* baru, model mampu beradaptasi dengan kelas atau variasi data baru, sehingga meningkatkan kinerjanya pada tugas deteksi objek tertentu.

Teknik ini umum digunakan dalam aplikasi visi komputer untuk mencapai akurasi yang lebih baik dengan jumlah data dan waktu pelatihan yang lebih sedikit (He et al., 2018). Selain itu, *transfer learning* dalam deteksi objek juga dapat membantu mengatasi masalah data terbatas yang sudah dianotasi (Vasconcelos et al., 2022). Ketika menggunakan YOLOv8 untuk

transfer learning dalam deteksi objek, penting untuk mempertimbangkan bobot yang digunakan dalam proses tersebut. Model YOLOv8 yang telah dilatih sebelumnya dilengkapi dengan *pretrain* bobotnya sendiri yang diperoleh dari *dataset* Microsoft COCO (Lin Tsung-Yiand Maire, 2014). Bobot ini berharga dalam mengekstrak fitur dari gambar dan dapat menjadi titik awal yang kuat untuk penyesuaian kembali pada *dataset* baru yang spesifik (Ganesh et al., 2021).

3.5 Evaluasi Model

Setelah model dilatih, model tersebut dievaluasi dengan beberapa metrik, seperti mAP (*mean Average Precision*). Metrik evaluasi ini membantu dalam menilai kinerja model deteksi objek pada *dataset* tertentu yang telah diatur ulang. Metrik mAP pada persamaan (2.5), memberikan ukuran komprehensif terhadap presisi dan *recall* model di berbagai kategori objek. Metrik lain seperti presisi, *recall*, dan F1-*score* juga dapat digunakan untuk mengevaluasi kinerja model dalam mendeteksi dan lokaliser objek pada gambar. Proses evaluasi sangat penting untuk memahami seberapa baik model YOLOv8 yang telah diatur ulang berkinerja pada *dataset* baru dan mengidentifikasi area-area yang dapat ditingkatkan.

3.6 Integrasi Model

Setelah model dilatih dan dievaluasi, langkah selanjutnya adalah melakukan integrasi model kedalam perangkat bergerak. Proses integrasi model kedalam perangkat bergerak dimulai dengan melakukan konversi model menjadi format yang sesuai dengan *library* yang akan digunakan pada perangkat bergerak lalu melakukan pengujian waktu inferensi antar berbagai jenis perangkat.

3.6.1 Konversi Model

Konversi model deteksi objek ke perangkat bergerak telah mengalami perkembangan signifikan selama bertahun-tahun, didorong oleh peningkatan permintaan akan deteksi objek yang efisien dan akurat dalam berbagai aplikasi, seperti kendaraan otonom, realitas virtual, dan manufaktur komersial. Dengan kemajuan teknologi pemindaian 3D yang cepat dan ketersediaan luas sensor 3D portabel, perangkat bergerak telah mampu untuk pengenalan objek 3D secara *real-time*, menyebabkan lonjakan dalam penelitian tentang algoritma deteksi objek 3D (Song & Guo, 2022).

Pendekatan awal untuk deteksi objek bergerak berfokus pada pemindahan tugas-tugas kompleks ke server cloud, tetapi strategi ini menghadapi tantangan karena latensi jaringan dan kapasitas jaringan yang terbatas (Chen et al., 2023). Sebagai hasilnya, para peneliti mulai menjelajahi model-model ringan dan teknik akselerasi perangkat keras untuk mengoptimalkan kinerja pada perangkat seluler. Sebagai contoh, MobileDenseNet dikembangkan untuk mengatasi masalah deteksi objek kecil dan lokalitas yang tidak akurat dalam deteksi objek satu tahap, mencapai akurasi tinggi sambil mempertahankan efisiensi *real-time* pada perangkat seluler (Hajizadeh et al., 2022).

Penelitian ini menggunakan format NCNN yang dikembangkan oleh Tencent karena memiliki performa yang baik dalam komputasi *neural network*. NCNN adalah *library* komputasi inferensi ANN berperforma tinggi yang dioptimalkan untuk platform perangkat bergerak. NCNN sangat memperhatikan penerapan dan penggunaan di perangkat bergerak sejak awal desain. NCNN tidak memiliki ketergantungan pada pihak ketiga. Ini bersifat lintas platform dan berjalan lebih cepat daripada semua *library* terbuka yang diketahui di CPU perangkat bergerak. *Library* memungkinkan untuk melakukan pengubahan model kedalam aplikasi perangkat bergerak dengan efisien. Saat ini, NCNN digunakan dalam banyak aplikasi Tencent, seperti QQ, Qzone, WeChat, Pitu, dan sebagainya(Ni & The ncnn contributors, 2017).

NCNN lalu dikembangkan untuk melakukan inferensi dengan perangkat bergerak dengan model YOLOv8 dalam ncnn-android-yolov8 (FeiGeChuanShu & Q-Engineering, 2023) dengan menggunakan Huawei Nova 10 Pro. Integrasi model YOLOv8 dilakukan dengan mengubah format model yang telah dikembangkan diekspor dari PyTorch (pt) ke *Open Neural Network Exchange* (ONNX) dan ke NCNN.

ONNX, atau Open Neural Network Exchange, dirancang sebagai format perantara antara berbagai model pembelajaran mesin yang memungkinkan interoperabilitas dan portabilitas model di berbagai aplikasi dan *library*. ONNX memfasilitasi konversi model dengan arsitektur yang berbeda, memberikan fleksibilitas kepada pengembang untuk memilih arsitektur terbaik untuk setiap tahap dari pengembangan model hingga implementasi (ONNX, 2024). Di sisi lain, NCNN adalah kerangka kerja neural network yang sangat dioptimalkan untuk perangkat mobile, khususnya untuk CPU ponsel. NCNN dirancang dengan fokus pada kecepatan dan efisiensi, memungkinkan model pembelajaran mesin untuk berjalan lebih cepat dan lebih hemat daya dibandingkan dengan kerangka kerja lainnya yang dikenal di CPU ponsel. Dengan optimisasi khusus yang diterapkan pada NCNN, aplikasi-aplikasi berbasis AI dapat

memberikan performa yang lebih baik dan responsif di perangkat mobile, sehingga meningkatkan pengalaman pengguna secara keseluruhan.

3.6.2 *Tipe Perangkat*

Jenis perangkat yang digunakan merupakan faktor penting yang perlu dipertimbangkan dalam deteksi objek pada perangkat bergerak. Dalam hal ini, Samsung A6 dengan prosesor Samsung Exynos 7870 octa-core 1,6 GHz dan RAM 3GB, serta Samsung A54 dengan RAM 6GB dan prosesor Octa-core (4x2,4 GHz Cortex-A78 & 4x2,0 GHz Cortex-A55), telah dievaluasi. Selain mengevaluasi kinerja model pada perangkat bergerak dengan menggunakan model FP-16 dan INT-8 pada Samsung A6 dan A54, langkah selanjutnya adalah membandingkan langsung kinerja proses pelatihan mereka dengan perangkat lain. Untuk melakukan perbandingan komprehensif *mean Average Precision* (mAP) dan waktu inferensi, penelitian ini juga menggunakan NVIDIA Tesla V100.

Perbandingan model FP-32 ini memberikan pemahaman yang komprehensif terhadap kinerja selama proses pelatihan, termasuk evaluasi bobot yang dihasilkan dari pelatihan pada berbagai model dan konfigurasi perangkat keras. Ini memungkinkan perbandingan kinerja pada perangkat bergerak dengan kinerja pada GPU menggunakan NVIDA Tesla V100. Metrik utama yang digunakan untuk perbandingan adalah *mean Average Precision* (mAP) dan waktu inferensi. Dengan membandingkan metrik ini di berbagai perangkat dan jenis model, kita dapat memperoleh wawasan tentang kinerja keseluruhan dari model deteksi objek.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Pengumpulan *Dataset*

Pengumpulan data dilakukan dengan mempertimbangkan kriteria seleksi toko ritel, metode pengambilan foto secara langsung di lapangan, dan dari hasil tersebut telah didaptkan 11.216 foto. Melalui kunjungan langsung ke berbagai toko ritel yang dipilih, berhasil diambil foto-foto dari berbagai produk, dan kondisi di dalam toko yang sesuai dengan aturan pengambilan foto. Manajemen data dilakukan dengan menyimpan foto-foto dalam struktur folder yang terorganisir dan menjaga privasi serta kerahasiaan informasi dari toko ritel tersebut. Beberapa contoh foto dapat dilihat seperti pada Gambar 4.1 dan Gambar 4.2.

Gambar 4.1 Contoh Pengambilan Foto Rak

Gambar 4.2 Contoh Pengambilan Foto Rak

4.2 Pemrosesan Awal Data

Proses pemrosesan awal data melibatkan *resizing* pada setiap gambar dalam *dataset*. Gambar-gambar sebelumnya memiliki dimensi yang beragam, seperti 4080x3060, 1024x768, dan 2040x1530. Dalam rangka mengoptimalkan ukuran gambar sesuai dengan kebutuhan komputasi pada pengembangan model deteksi objek, dilakukan resizing menjadi dimensi 1024x1024. Pilihan dimensi tersebut didasarkan pada upaya mencapai keseimbangan antara akurasi deteksi dan efisiensi komputasi. Selain itu, *resizing* ke dimensi 1024x1024 dipilih untuk menyederhanakan representasi visual objek tanpa kehilangan informasi kritis, memungkinkan model beroperasi secara efisien pada sumber daya komputasi yang terbatas.

4.3 Pelabelan *Dataset*

Pelabelan *dataset* dilakukan dengan cara memuat setiap gambar ke dalam perangkat lunak LabelImg. Kemudian, objek-objek yang ingin dilabeli diidentifikasi dan diberi anotasi dengan menarik *bounding box* di sekitar objek tersebut. Setelah selesai melabeli satu gambar,

dilanjutkan dengan gambar berikutnya hingga semua gambar dalam *dataset* terlabeli dengan baik. Proses pelabelan ini membutuhkan kecermatan dan akurasi untuk memastikan setiap objek teridentifikasi dengan benar. Setelah selesai, LabelImg menyimpan informasi label dalam format YOLO yang sesuai dengan standar yang dapat dibaca oleh *framework* YOLO. *Dataset* yang telah dilabeli siap untuk digunakan dalam pelatihan model deteksi objek, membentuk dasar yang penting untuk pengembangan model yang akurat dan andal. Gambar 4.3 menunjukkan *dataset* yang siap untuk dilatih menggunakan YOLOv8. Dari gambar tersebut, dapat dilihat bahwa terdapat beberapa baris yang berisikan sebuah angka numerik dan 4 angka decimal. Setiap baris pada gambar tersebut merupakan anotasi dari sebuah objek pada gambar. Angka numerik pada kolom 1 menunjukkan angka penomoran pada kelas dataset yang digunakan, sedangkan 4 angka decimal tersebut merupakan koordinat dari titik kiri atas sebuah objek dan titik kanan bawah dari sebuah objek yang berbentuk persegi panjang.

```
(yolov8) jupyter-20523239@cds-tesla:~/yolov8/data/datasets/data_skripsi/val$ cat val_0_36.txt
21 0.1608072966337204 0.3369140625 0.1471354067325592 0.169921875
21 0.306640625 0.3447265625 0.1471354216337204 0.169921875
21 0.462890625 0.361328125 0.1471354067325592 0.169921875
9 0.6067708730697632 0.37353515625 0.1510416865348816 0.1533203125
47 0.7662760019302368 0.388671875 0.1653645634651184 0.15625
47 0.91796875 0.39794921875 0.1640625 0.1533203125
20 0.1940104216337204 0.55810546875 0.1536458432674408 0.1435546875
2 0.3502604365348816 0.5751953125 0.1588541567325592 0.15234375
19 0.5188802480697632 0.58740234375 0.1627604365348816 0.1533203125
15 0.68359375 0.59521484375 0.16927087306976318 0.1533203125
3 0.8548176884651184 0.60986328125 0.18359375 0.1630859375
```

Gambar 4.3 *Dataset* YOLOv8

4.4 Pelatihan Model

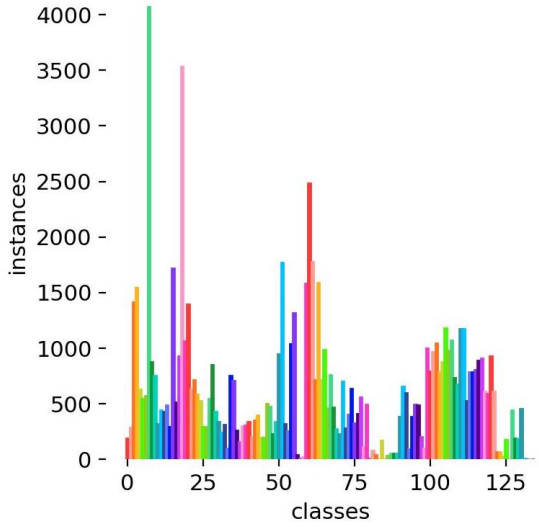
Pelatihan model YOLOv8 melibatkan beberapa tahapan penting untuk memastikan model dapat mengenali objek dengan akurasi tinggi. Proses ini dimulai dengan membagi dataset menjadi tiga bagian, yaitu data latih dan data validasi. Data latih digunakan untuk mengajarkan model dan data validasi untuk mengukur performa selama pelatihan dan menghindari overfitting.

4.4.1 Split *Dataset*

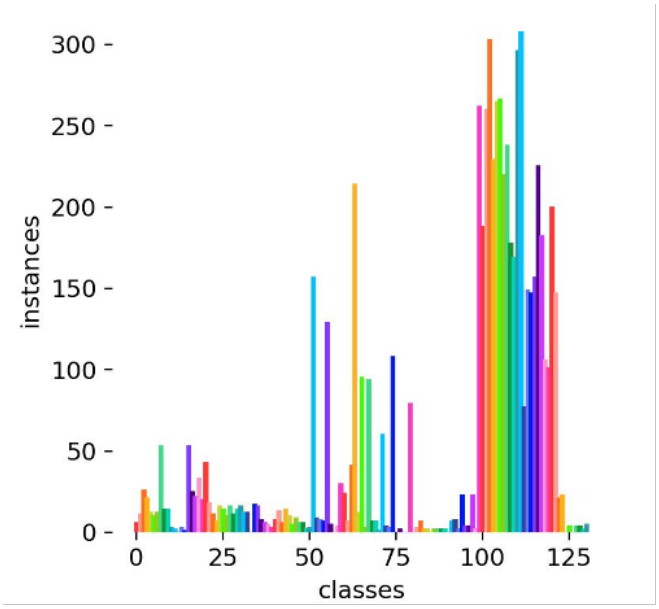
Setelah pengumpulan dan pelabelan *dataset* menggunakan LabelImg, langkah selanjutnya adalah membagi *dataset* menjadi subset pelatihan (*train*) dan validasi (*test*) dengan proporsi 90:10. Proses ini penting untuk menguji kinerja model pada data yang belum pernah dilihat sebelumnya. Pertama, *dataset* dipisahkan menjadi dua kelompok, yaitu data pelatihan yang akan digunakan untuk melatih model dan data validasi yang akan digunakan untuk

menguji performa model. Selanjutnya, data dipilih secara acak dan ditempatkan dalam dua kelompok tersebut dengan mempertahankan proporsi 90% untuk data pelatihan dan 10% untuk data validasi. Dari hasil pembagian *dataset* tersebut, telah didapatkan data *train* sebanyak 10601 dan data *test* sebanyak 615 foto.

Data pada gambar pelatihan memiliki 59.235 objek dengan distribusi seperti Gambar 4.4, sedangkan pada gambar validasi memiliki 6.619 objek dengan distribusi seperti Gambar 4.5.



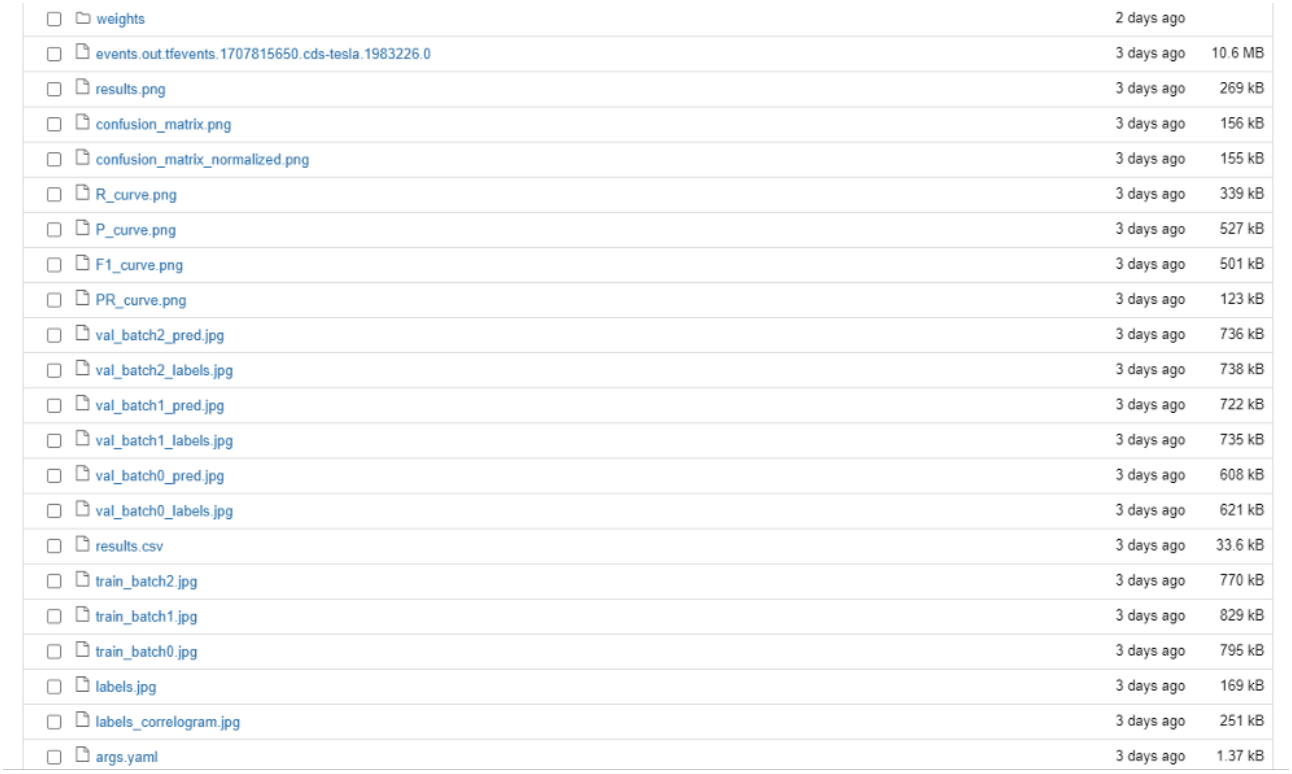
Gambar 4.4 Distribusi *Dataset* Latih



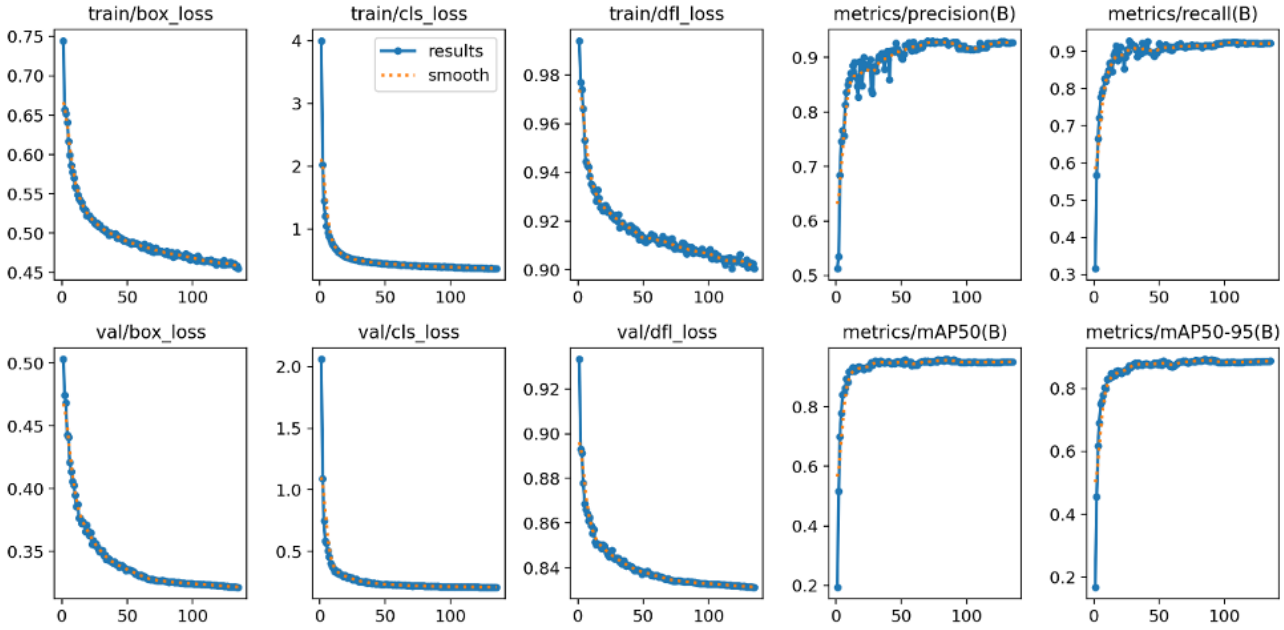
Gambar 4.5 Distribusi *Dataset* Evaluasi

4.4.2 Hasil *Training* YOLOv8n

Proses *training* YOLOv8n menghasilkan berbagai file keluaran yang dapat dilihat pada Gambar 4.6. Hasil tersebut memberikan gambaran menyeluruh mengenai performa dan kemampuan model yang telah dilatih. *Log training* mencatat progres pembelajaran, termasuk *epoch*, *loss*, dan waktu yang dibutuhkan. File *weights*, biasanya dalam format .pt menyimpan model yang sudah terlatih dan siap digunakan untuk deteksi objek pada gambar atau video baru. Selain itu, hasil evaluasi yang diukur melalui metrik seperti mAP, *precision*, dan *recall*, memberikan indikasi tingkat keakuratan dan kecepatan model dalam mendeteksi objek. Terakhir, visualisasi pada Gambar 4.7 yang menampilkan kotak pembatas di sekitar objek yang terdeteksi, membantu visualisasi performa model secara langsung.



Gambar 4.6 Hasil Pelatihan YOLOv8n



Gambar 4.7 Visualisasi Pelatihan YOLOv8n

Hasil *training* YOLOv8n menunjukkan bahwa model mengalami peningkatan performa selama proses training. Hal ini dapat dilihat dari grafik yang menunjukkan penurunan nilai *loss training* dan validation *loss*. Nilai box *loss training* turun dari 0.75 menjadi 0.45, sedangkan nilai box *loss* validation turun dari 0.5 menjadi kurang dari 0.35. Penurunan nilai *loss* ini menunjukkan bahwa model semakin baik dalam memprediksi *bounding box* yang tepat untuk objek dalam gambar. Sedangkan pada nilai *classification loss* pada *training* turun dari nilai 4.0 hingga kurang dari 1.0 dan pada validation turun dari 2.0 hingga kurang dari 0.5. Penurunan nilai ini menunjukkan bahwa model semakin baik dalam memprediksi label kelas. Nilai DFL *loss* atau (*Distribution Focal Loss*) *training* turun dari 0.98 hingga 0.9 dan pada *validation* turun dari 0.92 hingga 0.84. Penurunan nilai ini menunjukkan bahwa model semakin baik dalam melakukan pembobotan dalam deteksi objek karena *loss* ini bertujuan untuk melakukan pembobotan saat kelas terdapat imbalansi.

Selain itu, nilai mAP50, *precision* dan *recall* model juga mengalami peningkatan. Nilai mAP50 naik dari 0.2 hingga 0.8. Nilai *precision* naik dari 0.4 menjadi 0.9, sedangkan nilai *recall* naik dari 0.3 menjadi 0.9. *Precision* menunjukkan proporsi prediksi positif yang benar, sedangkan *recall* menunjukkan proporsi objek positif yang terdeteksi. Peningkatan kedua nilai ini menunjukkan bahwa model semakin akurat dalam mendeteksi objek dan semakin kecil kemungkinan untuk memberikan prediksi positif yang salah. Model ini dilatih dengan 300

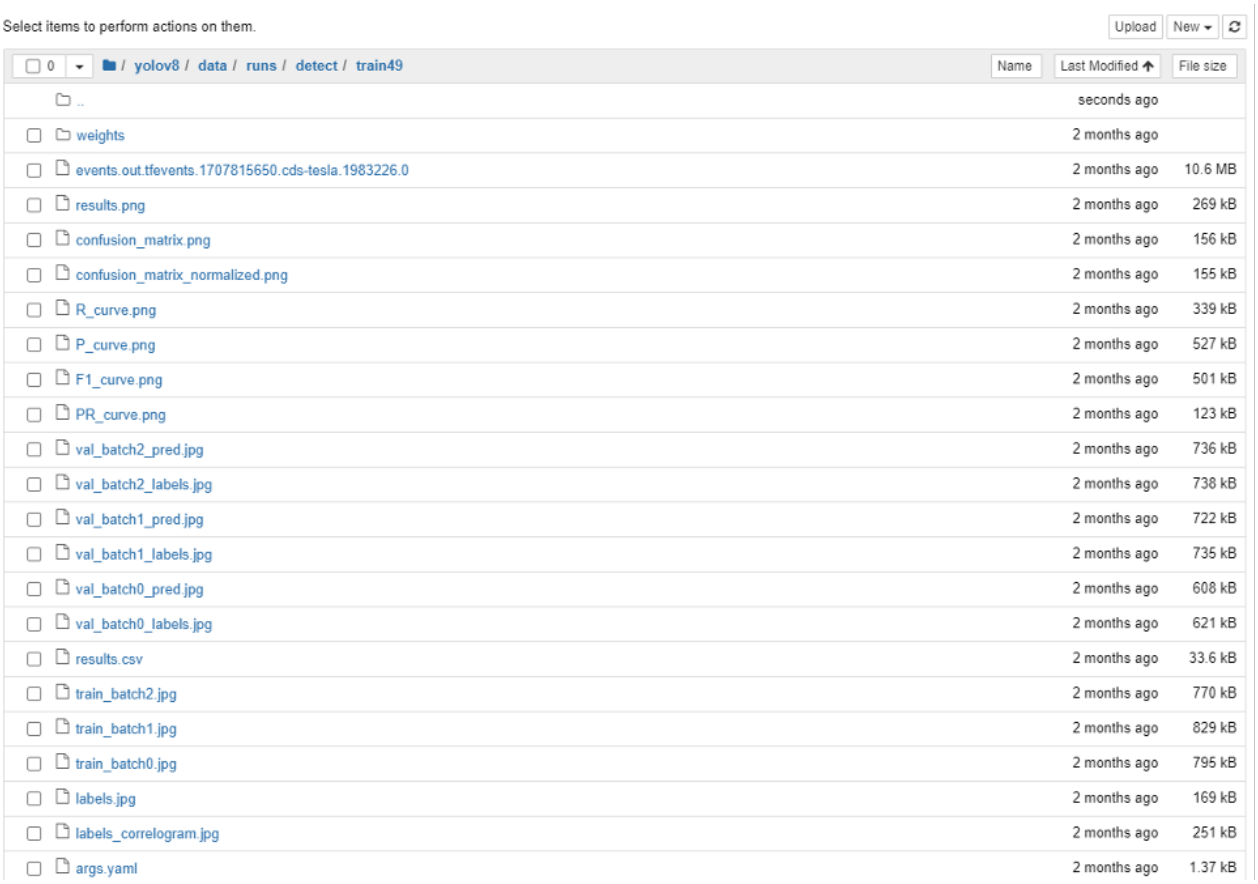
epoch tetapi berhenti pada 125 *epoch*. Hal ini terjadi karena tidak ada penurunan *loss* setelah *epoch* tersebut.

YOLOv8n setelah pelatihan menghasilkan file best.pt yang menyimpan bobot dari YOLOv8n dan berukuran 6.93 MB. Selain itu, YOLOv8n ini memiliki 168 layer, 3351329 parameter, 19 ms waktu inferensi dan 9.7 GFLOPs (*Giga Floating Operations Per Seconds*).

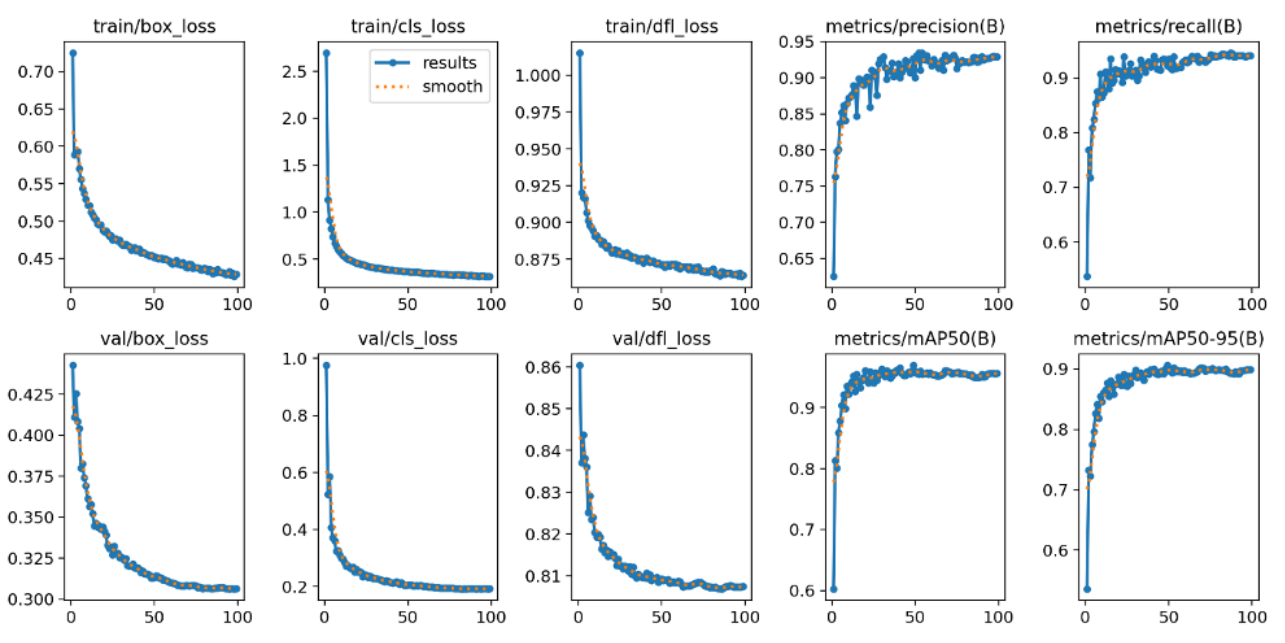
Secara keseluruhan, hasil *training* menunjukkan bahwa model YOLOv8n telah dilatih dengan baik dan mampu mendeteksi objek dengan performa yang cukup baik. Peningkatan nilai *loss*, *precision*, dan recall menunjukkan bahwa model semakin akurat dalam memprediksi label kelas dan *bounding box* yang tepat untuk objek dalam gambar.

4.4.3 Hasil *Training* YOLOv8s

Proses *training* YOLOv8s menghasilkan berbagai file *output* yang sama dengan YOLOv8n. *Output* dari proses *training* dapat dilihat pada Gambar 4.8.



Gambar 4.8 Hasil Pelatihan YOLOv8s



Gambar 4.9 Visualisasi Pelatihan YOLOv8s

Hasil *training* YOLOv8s menunjukkan bahwa model mengalami peningkatan performa selama proses *training*. Hal ini dapat dilihat dari grafik yang menunjukkan penurunan nilai *loss training* dan *validation loss*. Nilai *box loss training* turun dari 0.7 menjadi 0.45, sedangkan nilai *box loss validation* turun dari 0.425 menjadi 0.3. Penurunan nilai *loss* ini menunjukkan bahwa model semakin baik dalam memprediksi *bounding box* yang tepat untuk objek dalam Gambar 4.9. Sedangkan pada nilai *classification loss* pada *training* turun dari nilai 2.5 hingga 0.5 dan pada *validation* turun dari 1.0 hingga kurang dari 0.2. Penurunan nilai ini menunjukkan bahwa model semakin baik dalam memprediksi label kelas. Nilai DFL *loss* atau (*Distribution Focal Loss*) *training* turun dari 1.0 hingga 0.875 dan pada *validation* turun dari 0.86 hingga 0.81. Penurunan nilai ini menunjukkan bahwa model semakin baik dalam melakukan pembobotan dalam deteksi objek karena *loss* ini bertujuan untuk melakukan pembobotan saat kelas terdapat imbalansi.

Selain itu, model menunjukkan peningkatan performa yang cukup baik, dibuktikan dengan kenaikan nilai metrik evaluasi utama. Nilai mAP50 melonjak dari 0.6 menjadi 0.9, *precision* meningkat dari 0.65 menjadi 0.95, dan *recall* naik dari 0.6 menjadi 0.9. Perbaikan ini menunjukkan bahwa model semakin akurat dalam mengidentifikasi objek dan meminimalisir prediksi positif yang keliru. Meskipun model dilatih dengan 300 *epoch*, proses pelatihan dihentikan pada epoch ke-110. Pemberhentian ini terjadi karena tidak ada penurunan loss yang signifikan setelahnya.

Setelah proses pelatihan, YOLOv8s menghasilkan file bernama "best.pt" yang memuat bobot model dengan ukuran 22.6 MB. Model YOLOv8s ini memiliki arsitektur yang terdiri dari 168 layer dan 11.177.829 parameter. Model ini mampu menghasilkan prediksi dengan waktu inferensi 19.9 ms dan memiliki kompleksitas komputasi sebesar 28.7 GFLOPs (*Giga Floating Operations Per Seconds*).

Secara keseluruhan, proses pelatihan menunjukkan performa YOLOv8s yang memuaskan dalam mendeteksi objek. Hal ini dibuktikan dengan penurunan nilai *loss*, serta peningkatan nilai *precision*, dan *recall*, yang menunjukkan peningkatan akurasi model dalam memprediksi label kelas dan bounding box yang tepat untuk objek dalam gambar.

4.5 Evaluasi Model

Evaluasi model YOLOv8 dilakukan dengan menggunakan data validasi yang telah dipisahkan sebelumnya, terdiri dari 615 gambar dan 6619 objek. Evaluasi ini dilakukan dengan memanfaatkan GPU NVIDIA Tesla V100. Hasil evaluasi ini dirangkum dalam Tabel 4.1. Lalu, untuk hasil inferensi model YOLOv8n dan YOLOv8s dapat dilihat pada Gambar 4.10.

Tabel 4.1 Tabel Hasil Evaluasi Model YOLOv8

Model	**Waktu Inferensi**	**mAP50**	**mAP50:95**	***Presisi***	***Recall***
YOLOV8n	19 ms	0.961	0.895	0.922	0.918
YOLOV8s	19.9 ms	0.969	0.905	0.92	0.927

YOLOv8n Label YOLOv8s

Gambar 4.10 Hasil Deteksi dengan YOLOv8n dan YOLOv8s

Selain itu, terdapat hasil evaluasi dari YOLOv8s yang dirangkum berdasarkan kelas dan jumlah objeknya. Evaluasi ini memberikan gambaran yang jelas mengenai performa model dalam mendeteksi berbagai kelas objek. Hasil-hasil tersebut sangat penting untuk memahami kekuatan dan kelemahan YOLOv8s dalam tugas deteksi objek. Hasil evaluasi tersebut dapat dilihat pada Tabel 4.2.

Tabel 4.2 Hasil Evaluasi YOLOv8s

Kelas	**AP50**	***Instances***
NESTLE BATITA 1+ Madu+Iron 40x150g N1 ID-12070913-00300900100160140030001	0.995	6
NESTLE BATITA 1+ Madu+Iron 24x400g ID-12362980-00300900100160140030001	0.9059655562	11
LACTOGEN 2 Happynutri 12x750g N1 ID_REDESIGN1	0.995	26
LACTOGROW 4 Happynutri Honey12x750g N1ID_REDESIGN1	0.9886363636	21
LACTOGROW 4 Happynutri Van 12x750g N1 ID_REDESIGN1	0.8944377622	12
LACTOGEN 2 Happynutri 40x180g N1 ID_REDESIGN1	0.995	10
LACTOGEN 1 Happynutri 40x180g N1 ID_REDESIGN1	0.8891733333	12
FRISIAN FLAG	0.9670687646	53
LACTOGROW 3 Happynutri Van 24x350g N1 ID_REDESIGN1	0.9903333333	14
LACTOGEN 2 Happynutri 24x350g N1 ID_REDESIGN1	0.9903333333	14
LACTOGEN 1 Happynutri 12x1kg ID_REDESIGN1	0.9125	3
LACTOGEN 2 Happynutri 12x1kg ID_REDESIGN1	0.995	2
LACTOGROW 3 Happynutri Van 12x1kg ID_REDESIGN1	0.863	3
LACTOGROW 4 Happynutri Van 12x1kg ID_REDESIGN1	0.995	1
LACTOGROW 3 Happynutri Honey 12x750gN1ID_REDESIGN1	0.9868565486	53
LACTOGROW 4 Happynutri Honey12x1kg ID_REDESIGN1	0.9907142857	25

LACTOGROW 3 Happynutri Van 12x750g N1 ID_REDESIGN1	0.9459268992	22
MORINAGA	0.9058090763	33
LACTOGROW 3 Happynutri 12x750g N1 ID_REDESIGN1	0.9854545455	20
LACTOGEN 1 Happynutri 12x750g N1 ID_REDESIGN1	0.995	43
LACTOGEN 1 Happynutri 24x350g N1 ID_REDESIGN1	0.9923684211	18
LACTOGROW 3 Happynutri Honey 24x350gN1ID_REDESIGN1	0.995	11
DANCOW 5+ Van Advn ExcNutr 24x400g ID-12010754-00300901001501300 9001_REDESIGN1	0.7558333333	7
DANCOW 5+ Madu Advn ExcNutr 24x400g ID-12012293-00300901001501300 9001_REDESIGN1	0.7444905713	16
DANCOW 5+ Cok Advn ExcNutr 24x400g ID-12011934-00300901001501300 9001_REDESIGN1	0.995	14
DANCOW 5+ Cok Advn ExcNutr 12x1000g ID-12210282-00300901001501300 9001_REDESIGN1	0.995	10
DANCOW 3+ Van Advn ExcNutr 12x800g ID-12005039-00300901001401200 9001_REDESIGN1	0.995	16
DANCOW 3+ Madu Advn ExcNutr 12x800g ID-12005038-00300901001401200 9001_REDESIGN1	0.95	11
DANCOW 3+ Cok Advn ExcNutr 12x800g ID-12006469-00300901001401200 9001_REDESIGN1	0.995	14
DANCOW 3+ Cok Advn ExcNutr 24x400g ID-12006468-00300901001401200 9001_REDESIGN1	0.995	16
DANCOW 5+ Cok Advn ExcNutr 12x800g ID-12011935-00300901001501300 9001_REDESIGN1	0.995	12
DANCOW 5+ Madu Advn ExcNutr 12x800g ID-12012200-00300901001501300 9001_REDESIGN1	0.948	12
DANCOW 1+ Madu Advn ExcNutr 12x800g ID-12005045-00300901001301100 9001_REDESIGN1	0.9190735786	17

DANCOW 5+ Van Advn ExcNutr 12x800g ID-12010738-00300901001501300900l_REDESIGN1	0.789474359	16
DANCOW 1+ Van Advn ExcNutr 12x800g N1 ID-12004999-00300901001301100900l_REDESIGN1	0.995	8
DANCOW 1+ Cok Advn ExcNutr 12x800g N1 ID-12010753-00300901001301100900l_REDESIGN1	0.995	6
DANCOW 1+ Van Advn ExcNutr 24x400g ID-12005037-00300901001301100900l_REDESIGN1	0.995	5
DANCOW 1+ Madu Advn ExcNutr 24x400g N1ID-12004998-00300901001301100900l_REDESIGN1	0.995	3
DANCOW 3+ Madu Advn ExcNutr 40x200g ID-12005050-00300901001401200900l_REDESIGN1	0.9435714286	8
NESTLE DATITA 3+ Madu+Iron 24x400g ID-12363044-00300901001701900300l	0.9842857143	13
DANCOW 3+ Van Advn ExcNutr 24x400g ID-12005052-00300901001401200900l_REDESIGN1	0.995	6
DANCOW 3+ Madu Advn ExcNutr 24x400g ID-12005051-00300901001401200900l_REDESIGN1	0.6976523396	14
NESTLE BATITA 1+Vanilla+Iron 12x900g ID-12383230-00300901001601400300l	0.995	10
NESTLE DATITA 3+ Madu+Iron 12x900g ID-12383207-00300901001701900300l	0.9378571429	5
LACTOGROW 3 Happynutri Honey 40x180gN1ID_REDESIGN1	0.8933368421	9
LACTOGROW 3 Happynutri 24x350g N1 ID_REDESIGN1	0.8983333333	6
DANCOW 1+ Cok Advn ExcNutr 24x400g N1 ID-12010752-00300901001301100900l_REDESIGN1	0.995	6
DANCOW 5+ Van Advn ExcNutr 24x400g ID_REDESIGN2	0.4975	2
NESTLE BATITA 1+ Madu+Iron 12x850g ID	0.995	3

DANCOW 1+ Madu Advn ExcNutr 12x800g ID_REDESIGN2	0.9939651669	157
NESTLE BATITA 1+ Vanilla+Iron 24x400g ID-12363042-00300901001601400 3001	0.8921923077	9
NESTLE BATITA 1+ Madu+Iron 12x900g ID-12383231-00300901001601400 3001	0.995	8
DANCOW 3+ Madu Advn ExcNutr 12x800g ID_REDESIGN2	0.995	7
DANCOW 5+ Madu Advn ExcNutr 12x800g ID_REDESIGN2	0.9852872841	129
DANCOW 5+ Madu Advn ExcNutr 12x1000g ID-12210283-00300901001501300 9001_REDESIGN1	0.995	5
NESTLE DATITA 3+ Madu+Iron 40x150g N1 ID-12147994-00300901001701900 3001	0.995	4
SGM	0.9790690233	30
VIDORAN	0.9919230769	24
ENFAGROW	0.883	7
DANCOW 1+ Van Advn ExcNutr 12x800g N1 ID_REDESIGN2	0.9804068505	41
DANCOW 3+ Van Advn ExcNutr 12x800g ID_REDESIGN2	0.9807271164	214
DANCOW 3+ Madu Advn ExcNutr 12x1000g ID_REDESIGN2	0.9161904762	12
DANCOW 5+ Van Advn ExcNutr 12x800g ID_REDESIGN2	0.9866049039	95
DANCOW 3+ Cok Advn ExcNutr 12x800g ID_REDESIGN2	0.995	3
DANCOW 3+ Van Advn ExcNutr 24x400g ID_REDESIGN2	0.9892597518	94
DANCOW 5+ Madu Advn ExcNutr 24x400g ID_REDESIGN2	0.748	7
DANCOW 3+ Van Advn ExcNutr 12x1000g ID_REDESIGN2	0.7304	7

DANCOW 1+ Van Advn ExcNutr 12x1000g ID_REDESIGN2	0.4975	1
DANCOW 5+ Cok Advn ExcNutr 12x800g ID_REDESIGN2	0.995	60
DANCOW 1+ Madu Advn ExcNutr 12x1000gN1ID_REDESIGN2	0.8878571429	4
DANCOW 5+ Madu Advn ExcNutr 12x1000g ID_REDESIGN2	0.995	3
DANCOW 1+ Van Advn ExcNutr 24x400g ID_REDESIGN2	0.9930555556	108
DANCOW 1+ Madu Advn ExcNutr 24x400g N1ID_REDESIGN2	0.8283333333	2
DANCOW 1+ Cok Advn ExcNutr 12x800g N1 ID_REDESIGN2	0.99225	79
NESTLE DATITA 3+Vanilla+Iron 12x900g ID-12383206-00300901001701900 3001	0.995	3
DANCOW 1+ Cok Advn ExcNutr 12x800g N1 ID-12010753-00300901001301100 9001	0.995	2
DANCOW 1+ Madu Advn ExcNutr 12x1000gN1ID-12210289-00300901001301100 9001_REDESIGN1	0.995	2
DANCOW 3+ Van Advn ExcNutr 12x1000g ID-12210284-00300901001401200 9001_REDESIGN1	0.3475	2
LACTOGROW 4 Happynutri Van 12x1kg ID	0.995	2
LACTOGROW 4 Happynutri Honey12x1kg ID	0.995	2
LACTOGROW 3 Happynutri Van 12x1kg ID	0.995	2
LACTOGEN 1 Happynutri 40x180g N1 ID_REDESIGN2	0.9775	7
LACTOGEN 1 Happynutri 12x750g N1 ID_REDESIGN2	0.995	8
LACTOGEN 1 Happynutri 24x350g N1 ID_REDESIGN2	0.995	2
LACTOGEN 2 Happynutri 12x750g N1 ID_REDESIGN2	0.995	23
LACTOGROW 3 Happynutri Honey 12x750gN1ID_REDESIGN2	0.995	4

LACTOGROW 4 Happynutri Honey12x750g N1ID_REDESIGN2	0.995	4
LACTOGROW 3 Happynutri 12x750g N1 ID_REDESIGN2	0.995	23
LACTOGEN 2 Happynutri 24x350g N1 ID_REDESIGN2	0.995	2
S-26 PROCAL GOLD Can 24x400g N8 ID	0.9908088493	262
S-26 PROMIL GOLD 2 Can 24x400g N8 ID	0.9949470899	188
S-26 PROMIL GOLD 1 Can 24x400g N8 ID	0.993892014	260
S-26 ULTIMA PROMIL 1 Can Top 6x850g ID	0.9938245264	303
S-26 ULTIMA PROMIL 2 Can Top 6x850g ID	0.995	229
S-26 ULTIMA PROCAL Can Top 6x850g ID	0.9946226415	265
S-26 GOLD PROMIL 1 Can Top 6x900g N8 ID	0.9852664533	266
S-26 PROCAL GOLD Can Top 6x900g N8 ID	0.9938006382	220
S-26 PROMISE GOLD Can Top 6x900g N8 ID	0.990612035	238
S-26 PROMISE Pouch 12x400g N4 ID_REDESIGN1	0.9929733924	178
S-26 PROCAL Honey Pouch 12x400g ID	0.9944694451	169
S-26 PROMIL 1 Pouch 12x700g ID	0.9948989899	296
S-26 PROMIL 2 Pouch 12x700g ID	0.9949021702	308
S-26 PROCAL Pouch 12x400g N4 ID	0.9933333333	77
S-26 PROMIL 1 Pouch 12x400g N2 ID	0.9884236412	149
S-26 PROMIL 2 Pouch 12x400g N2 ID	0.993646359	147
S-26 PROCAL Pouch 6(2x700g) N2 ID	0.9904749018	157
S-26 PROMISE Pouch 12x700g N4 ID_REDESIGN1	0.9949118943	225
S-26 PROCAL Pouch 12x700g N4 ID	0.9948907104	182
S-26 PROCAL GOLD Can 6x1.6kg N8 ID	0.9737955775	106
S-26 PROMISE GOLD Can 6x1.6kg N8 ID	0.9727675585	101
S-26 GOLD PROMIL 2 Can Top 6x900g N8 ID	0.9911323462	200
S-26 PROMISE Pouch 6(2x700g) N1 ID	0.995	147
LACTOGROW 4 Happynutri Van 12x1kg ID_REDESIGN2	0.995	21

LACTOGROW 3 Happynutri Van 12x750g N1 ID_REDESIGN2	0.995	23
DANCOW 1+ Van Advn ExcNutr 12x1000g ID-12210287-00300901001301100 9001_REDESIGN1	0.9116666667	4
DANCOW 1+ Van Advn ExcNutr 40x200g N1 ID-12245505-00300901001301100 9001_REDESIGN1	0.995	4
S-26 PROMISE Pouch 12x400g N4 ID	0.995	4
S-26 PROMISE Pouch 12x700g N4 ID	0.995	2
NUTRICIA	0.2008566434	1

Hasil evaluasi juga dilakukan pada model YOLOv8n. Evaluasi yang dilakukan pada YOLOv8n menggunakan metrik yang sama dengan YOLOv8s. Metrik yang digunakan yaitu *average precission* dan juga *instances*. Hasil dari evaluasi dapat dilihat pada Tabel 4.3.

Tabel 4.3 Hasil Evaluasi YOLOv8n

Class	**AP50**	***Instances***
NESTLE BATITA 1+ Madu+Iron 40x150g N1 ID-12070913-00300901001601400 3001	0.995	6
NESTLE BATITA 1+ Madu+Iron 24x400g ID-12362980-0030090100160140030 01	0.923	11
LACTOGEN 2 Happynutri 12x750g N1 ID_REDESIGN1	0.995	26
LACTOGROW 4 Happynutri Honey12x750g N1ID_REDESIGN1	0.9518181818	21
LACTOGROW 4 Happynutri Van 12x750g N1 ID_REDESIGN1	0.8911111111	12
LACTOGEN 2 Happynutri 40x180g N1 ID_REDESIGN1	0.995	10
LACTOGEN 1 Happynutri 40x180g N1 ID_REDESIGN1	0.8985104895	12
FRISIAN FLAG	0.9764231951	53
LACTOGROW 3 Happynutri Van 24x350g N1 ID_REDESIGN1	0.9281470588	14

LACTOGEN 2 Happynutri 24x350g N1 ID_REDESIGN1	0.995	14
LACTOGEN 1 Happynutri 12x1kg ID_REDESIGN1	0.9125	3
LACTOGEN 2 Happynutri 12x1kg ID_REDESIGN1	0.995	2
LACTOGROW 3 Happynutri Van 12x1kg ID_REDESIGN1	0.995	3
LACTOGROW 4 Happynutri Van 12x1kg ID_REDESIGN1	0.995	1
LACTOGROW 3 Happynutri Honey 12x750gN1ID_REDESIGN1	0.9886551724	53
LACTOGROW 4 Happynutri Honey12x1kg ID_REDESIGN1	0.995	25
LACTOGROW 3 Happynutri Van 12x750g N1 ID_REDESIGN1	0.9488363636	22
MORINAGA	0.8699249277	33
LACTOGROW 3 Happynutri 12x750g N1 ID_REDESIGN1	0.9902380952	20
LACTOGEN 1 Happynutri 12x750g N1 ID_REDESIGN1	0.9928327561	43
LACTOGEN 1 Happynutri 24x350g N1 ID_REDESIGN1	0.9866666667	18
LACTOGROW 3 Happynutri Honey 24x350gN1ID_REDESIGN1	0.995	11
DANCOW 5+ Van Advn ExcNutr 24x400g ID-12010754-00300901001501300 9001_REDESIGN1	0.6912282051	7
DANCOW 5+ Madu Advn ExcNutr 24x400g ID-12012293-00300901001501300 9001_REDESIGN1	0.7725126437	16
DANCOW 5+ Cok Advn ExcNutr 24x400g ID-12011934-00300901001501300 9001_REDESIGN1	0.995	14
DANCOW 5+ Cok Advn ExcNutr 12x1000g ID-12210282-00300901001501300 9001_REDESIGN1	0.995	10
DANCOW 3+ Van Advn ExcNutr 12x800g ID-12005039-00300901001401200 9001_REDESIGN1	0.995	16
DANCOW 3+ Madu Advn ExcNutr 12x800g ID-12005038-00300901001401200 9001_REDESIGN1	0.98	11

DANCOW 3+ Cok Advn ExcNutr 12x800g ID-12006469-00300901001401200 9001_REDESIGN1	0.995	14
DANCOW 3+ Cok Advn ExcNutr 24x400g ID-12006468-00300901001401200 9001_REDESIGN1	0.995	16
DANCOW 5+ Cok Advn ExcNutr 12x800g ID-12011935-00300901001501300 9001_REDESIGN1	0.995	12
DANCOW 5+ Madu Advn ExcNutr 12x800g ID-12012200-00300901001501300 9001_REDESIGN1	0.9088782971	12
DANCOW 1+ Madu Advn ExcNutr 12x800g ID-12005045-00300901001301100 9001_REDESIGN1	0.9499828375	17
DANCOW 5+ Van Advn ExcNutr 12x800g ID-12010738-00300901001501300 9001_REDESIGN1	0.6846515873	16
DANCOW 1+ Van Advn ExcNutr 12x800g N1 ID-12004999-00300901001301100 9001_REDESIGN1	0.995	8
DANCOW 1+ Cok Advn ExcNutr 12x800g N1 ID-12010753-00300901001301100 9001_REDESIGN1	0.995	6
DANCOW 1+ Van Advn ExcNutr 24x400g ID-12005037-00300901001301100 9001_REDESIGN1	0.995	5
DANCOW 1+ Madu Advn ExcNutr 24x400g N1ID-12004998-00300901001301100 9001_REDESIGN1	0.995	3
DANCOW 3+ Madu Advn ExcNutr 40x200g ID-12005050-00300901001401200 9001_REDESIGN1	0.971	8
NESTLE DATITA 3+ Madu+Iron 24x400g ID-12363044-00300901001701900 3001	0.9856666667	13
DANCOW 3+ Van Advn ExcNutr 24x400g ID-12005052-00300901001401200 9001_REDESIGN1	0.995	6
DANCOW 3+ Madu Advn ExcNutr 24x400g ID-12005051-00300901001401200 9001_REDESIGN1	0.771	14
NESTLE BATITA 1+Vanilla+Iron 12x900g ID-12383230-00300901001601400 3001	0.995	10

NESTLE DATITA 3+ Madu+Iron 12x900g ID-12383207-00300901001701900 3001	0.9283333333	5
LACTOGROW 3 Happynutri Honey 40x180gN1ID_REDESIGN1	0.9675	9
LACTOGROW 3 Happynutri 24x350g N1 ID_REDESIGN1	0.9125	6
DANCOW 1+ Cok Advn ExcNutr 24x400g N1 ID-12010752-00300901001301100 9001_REDESIGN1	0.995	6
DANCOW 5+ Van Advn ExcNutr 24x400g ID_REDESIGN2	0.4975	2
NESTLE BATITA 1+ Madu+Iron 12x850g ID	0.995	3
DANCOW 1+ Madu Advn ExcNutr 12x800g ID_REDESIGN2	0.9948113208	157
NESTLE BATITA 1+ Vanilla+Iron 24x400g ID-12363042-00300901001601400 3001	0.8678489796	9
NESTLE BATITA 1+ Madu+Iron 12x900g ID-12383231-00300901001601400 3001	0.995	8
DANCOW 3+ Madu Advn ExcNutr 12x800g ID_REDESIGN2	0.9094444444	7
DANCOW 5+ Madu Advn ExcNutr 12x800g ID_REDESIGN2	0.9830916031	129
DANCOW 5+ Madu Advn ExcNutr 12x1000g ID-12210283-00300901001501300 9001_REDESIGN1	0.8807142857	5
NESTLE DATITA 3+ Madu+Iron 40x150g N1 ID-12147994-00300901001701900 3001	0.9116666667	4
SGM	0.9908064516	30
VIDORAN	0.995	24
ENFAGROW	0.8071428571	7
DANCOW 1+ Van Advn ExcNutr 12x800g N1 ID_REDESIGN2	0.9340511887	41

DANCOW 3+ Van Advn ExcNutr 12x800g ID_REDESIGN2	0.9762202749	214
DANCOW 3+ Madu Advn ExcNutr 12x1000g ID_REDESIGN2	0.8186904762	12
DANCOW 5+ Van Advn ExcNutr 12x800g ID_REDESIGN2	0.9763638898	95
DANCOW 3+ Cok Advn ExcNutr 12x800g ID_REDESIGN2	0.995	3
DANCOW 3+ Van Advn ExcNutr 24x400g ID_REDESIGN2	0.9946842105	94
DANCOW 5+ Madu Advn ExcNutr 24x400g ID_REDESIGN2	0.5456666667	7
DANCOW 3+ Van Advn ExcNutr 12x1000g ID_REDESIGN2	0.7456833333	7
DANCOW 1+ Van Advn ExcNutr 12x1000g ID_REDESIGN2	0.995	1
DANCOW 5+ Cok Advn ExcNutr 12x800g ID_REDESIGN2	0.995	60
DANCOW 1+ Madu Advn ExcNutr 12x1000gN1ID_REDESIGN2	0.8038235294	4
DANCOW 5+ Madu Advn ExcNutr 12x1000g ID_REDESIGN2	0.995	3
DANCOW 1+ Van Advn ExcNutr 24x400g ID_REDESIGN2	0.9946330275	108
DANCOW 1+ Madu Advn ExcNutr 24x400g N1ID_REDESIGN2	0.995	2
DANCOW 1+ Cok Advn ExcNutr 12x800g N1 ID_REDESIGN2	0.9947530864	79
NESTLE DATITA 3+Vanilla+Iron 12x900g ID-12383206-00300901001701900 3001	0.83	3

DANCOW 1+ Cok Advn ExcNutr 12x800g N1 ID-12010753-003009010013011009001	0.995	2
DANCOW 1+ Madu Advn ExcNutr 12x1000gN1ID-12210289-003009010013011009001_REDESIGN1	0.995	2
DANCOW 3+ Van Advn ExcNutr 12x1000g ID-12210284-003009010014012009001_REDESIGN1	0.695	2
LACTOGROW 4 Happynutri Van 12x1kg ID	0.995	2
LACTOGROW 4 Happynutri Honey12x1kg ID	0.995	2
LACTOGROW 3 Happynutri Van 12x1kg ID	0.995	2
LACTOGEN 1 Happynutri 40x180g N1 ID_REDESIGN2	0.995	7
LACTOGEN 1 Happynutri 12x750g N1 ID_REDESIGN2	0.8983333333	8
LACTOGEN 1 Happynutri 24x350g N1 ID_REDESIGN2	0.995	2
LACTOGEN 2 Happynutri 12x750g N1 ID_REDESIGN2	0.995	23
LACTOGROW 3 Happynutri Honey 12x750gN1ID_REDESIGN2	0.995	4
LACTOGROW 4 Happynutri Honey12x750g N1ID_REDESIGN2	0.995	4
LACTOGROW 3 Happynutri 12x750g N1 ID_REDESIGN2	0.995	23
LACTOGEN 2 Happynutri 24x350g N1 ID_REDESIGN2	0.995	2
S-26 PROCAL GOLD Can 24x400g N8 ID	0.9933992754	262
S-26 PROMIL GOLD 2 Can 24x400g N8 ID	0.9947354497	188
S-26 PROMIL GOLD 1 Can 24x400g N8 ID	0.9947358491	260
S-26 ULTIMA PROMIL 1 Can Top 6x850g ID	0.9914344127	303
S-26 ULTIMA PROMIL 2 Can Top 6x850g ID	0.99430131	229
S-26 ULTIMA PROCAL Can Top 6x850g ID	0.9943207547	265
S-26 GOLD PROMIL 1 Can Top 6x900g N8 ID	0.9845960121	266
S-26 PROCAL GOLD Can Top 6x900g N8 ID	0.9936398262	220
S-26 PROMISE GOLD Can Top 6x900g N8 ID	0.9921860215	238

S-26 PROMISE Pouch 12x400g N4 ID_REDESIGN1	0.9930337079	178
S-26 PROCAL Honey Pouch 12x400g ID	0.9928827175	169
S-26 PROMIL 1 Pouch 12x700g ID	0.9928328394	296
S-26 PROMIL 2 Pouch 12x700g ID	0.9949674267	308
S-26 PROCAL Pouch 12x400g N4 ID	0.9925842814	77
S-26 PROMIL 1 Pouch 12x400g N2 ID	0.9933892093	149
S-26 PROMIL 2 Pouch 12x400g N2 ID	0.9947963717	147
S-26 PROCAL Pouch 6(2x700g) N2 ID	0.9948093152	157
S-26 PROMISE Pouch 12x700g N4 ID_REDESIGN1	0.9866233915	225
S-26 PROCAL Pouch 12x700g N4 ID	0.9948913043	182
S-26 PROCAL GOLD Can 6x1.6kg N8 ID	0.9686328698	106
S-26 PROMISE GOLD Can 6x1.6kg N8 ID	0.9939045798	101
S-26 GOLD PROMIL 2 Can Top 6x900g N8 ID	0.9904325224	200
S-26 PROMISE Pouch 6(2x700g) N1 ID	0.9922972973	147
LACTOGROW 4 Happynutri Van 12x1kg ID_REDESIGN2	0.995	21
LACTOGROW 3 Happynutri Van 12x750g N1 ID_REDESIGN2	0.981071964	23
DANCOW 1+ Van Advn ExcNutr 12x1000g ID-12210287-00300901001301100 9001_REDESIGN1	0.7866666667	4
DANCOW 1+ Van Advn ExcNutr 40x200g N1 ID-12245505-00300901001301100 9001_REDESIGN1	0.895	4
S-26 PROMISE Pouch 12x400g N4 ID	0.995	4
S-26 PROMISE Pouch 12x700g N4 ID	0.995	2
NUTRICIA	0.2112	1

4.6 Integrasi Model

Setelah model selesai dievaluasi, selanjutnya model yang berbentuk PyTorchscript (.pt) akan dikonversi terlebih dahulu ke dalam format ONNX. Namun, sebelum melakukan konversi tersebut, perlu dilakukan perubahan baris kode pada *library ultralytics*, pada file

ultralyitics/nn/module/head.py pada fungsi *forward* dalam kelas *Detect* menjadi seperti pada Gambar 4.11.

```
def forward(self, x):
    shape = x[0].shape  # BCHW
    for i in range(self.nl):
        x[i] = torch.cat((self.cv2[i](x[i]), self.cv3[i](x[i])), 1)
    if self.training:
        return x
    elif self.dynamic or self.shape != shape:
        self.anchors, self.strides = (x.transpose(0, 1) for x in make_anchors(x, self.stride, 0.5))
        self.shape = shape

    # box, cls = torch.cat([xi.view(shape[0], self.no, -1) for xi in x], 2).split((self.reg_max * 4, self.nc), 1)
    # dbox = dist2bbox(self.dfl(box), self.anchors.unsqueeze(0), xywh=True, dim=1) * self.strides
    # y = torch.cat((dbox, cls.sigmoid()), 1)
    # return y if self.export else (y, x)

    print("ook")
    return torch.cat([xi.view(shape[0], self.no, -1) for xi in x], 2).permute(0, 2, 1)
```

Gambar 4.11 Kode Perubahan Fungsi *Forward*

Selain itu, perlu juga untuk merubah fungsi *forward_split* pada file *ultralytics/nn/module/block.py* . Fungsi tersebut terdapat pada kelas *C2f* dan diubah menjadi seperti Gambar 4.12.

```
def forward(self, x):
    # y = list(self.cv1(x).split((self.c, self.c), 1))
    # y.extend(m(y[-1]) for m in self.m)
    # return self.cv2(torch.cat(y, 1))

    print("ook")
    x = self.cv1(x)
    x = [x, x[:, self.c:, ...]]
    x.extend(m(x[-1]) for m in self.m)
    x.pop(1)
    return self.cv2(torch.cat(x, 1))
```

Gambar 4.12 Kode Perubahan Pada Fungsi *forward_split*

Hal ini dilakukan agar tensor *output* dari model *torch* lebih sederhana sehingga model yang akan dikonversi kompatibel dengan *library* NCNN yang akan digunakan. *Library ultralytics* yang digunakan adalah versi 8.0.162. Setelah itu, proses konversi model ke dalam ONNX dilakukan dengan menggunakan argumen *simplify=True* untuk melakukan optimisasi pada model dan *opset=13* untuk menggunakan operasi ONNX versi 13. Setelah itu format ONNX dirubah dalam format NCNN dan meghasilkan dua file yaitu bobot model yang memiliki ekstensi *.params* dan struktur model yang memiliki ekstensi *.bin* yang kemudian dapat dimasukkan ke dalam aplikasi bergerak (FeiGeChuanShu & Q-Engineering, 2023).

Setelah melakukan konversi model menjadi format NCNN, selanjutnya adalah melakukan installasi NCNN ke dalam platform untuk melakukan pengembangan perangkat bergerak. Setelah berhasil melakukan installasi NCNN, fungsi untuk membaca model YOLOv8 kedalam perangkat bergerak dapat dilihat seperti pada Gambar 4.13. Setelah dapat membaca model tersebut, langkah selanjutnya adalah melakukan inferensi model YOLOv8 dengan perangkat bergerak yang dikodekan seperti pada Gambar 4.14.

```
int Yolo::load(AAssetManager* mgr, const char* modeltype, int _target_size, const float* _mean_vals, const
float* _norm_vals, bool use_gpu)
{
    yolo.clear();
    blob_pool_allocator.clear();
    workspace_pool_allocator.clear();
```

```
    ncnn::set_cpu_powersave(2);
    ncnn::set_omp_num_threads(ncnn::get_big_cpu_count());

    yolo.opt = ncnn::Option();

#if NCNN_VULKAN
    yolo.opt.use_vulkan_compute = use_gpu;
#endif

    yolo.opt.num_threads = ncnn::get_big_cpu_count();
    yolo.opt.blob_allocator = &blob_pool_allocator;
    yolo.opt.workspace_allocator = &workspace_pool_allocator;

    char parampath[256];
    char modelpath[256];
    sprintf(parampath, "yolos.param", modeltype);
    sprintf(modelpath, "yolos.bin", modeltype);

    yolo.load_param(mgr, parampath);
    yolo.load_model(mgr, modelpath);

    target_size = _target_size;
    mean_vals[0] = _mean_vals[0];
    mean_vals[1] = _mean_vals[1];
    mean_vals[2] = _mean_vals[2];
    norm_vals[0] = _norm_vals[0];
    norm_vals[1] = _norm_vals[1];
    norm_vals[2] = _norm_vals[2];

    return 0;
}
```

Gambar 4.13 Kode untuk Membaca Model

```
int Yolo::detect(const cv::Mat& rgb, std::vector<Object>& objects, float prob_threshold, float
nms_threshold)
{
    int width = rgb.cols;
    int height = rgb.rows;
```

```
    // pad to multiple of 32
    int w = width;
    int h = height;
    float scale = 1.f;
    if (w > h)
    {
        scale = (float)target_size / w;
        w = target_size;
        h = h * scale;
    }
    else
    {
        scale = (float)target_size / h;
        h = target_size;
        w = w * scale;
    }

    ncnn::Mat in = ncnn::Mat::from_pixels_resize(rgb.data, ncnn::Mat::PIXEL_RGB2BGR, width, height, w,
h);

    // pad to target_size rectangle
    int wpad = (w + 31) / 32 * 32 - w;
    int hpad = (h + 31) / 32 * 32 - h;
    ncnn::Mat in_pad;
    ncnn::copy_make_border(in, in_pad, hpad / 2, hpad - hpad / 2, wpad / 2, wpad - wpad / 2,
ncnn::BORDER_CONSTANT, 0.f);

    in_pad.substract_mean_normalize(0, norm_vals);

    ncnn::Extractor ex = yolo.create_extractor();

    ex.input("images", in_pad);

    std::vector<Object> proposals;

    ncnn::Mat out;
    ex.extract("output0", out);

    std::vector<int> strides = {8, 16, 32}; // might have stride=64
```

```
    std::vector<GridAndStride> grid_strides;
    generate_grids_and_stride(in_pad.w, in_pad.h, strides, grid_strides);
    generate_proposals(grid_strides, out, prob_threshold, proposals);

    // sort all proposals by score from highest to lowest
    qsort_descent_inplace(proposals);

    // apply nms with nms_threshold
    std::vector<int> picked;
    nms_sorted_bboxes(proposals, picked, nms_threshold);

    int count = picked.size();

    objects.resize(count);
    for (int i = 0; i < count; i++)
    {
        objects[i] = proposals[picked[i]];

        // adjust offset to original unpadded
        float x0 = (objects[i].rect.x - (wpad / 2)) / scale;
        float y0 = (objects[i].rect.y - (hpad / 2)) / scale;
        float x1 = (objects[i].rect.x + objects[i].rect.width - (wpad / 2)) / scale;
        float y1 = (objects[i].rect.y + objects[i].rect.height - (hpad / 2)) / scale;

        // clip
        x0 = std::max(std::min(x0, (float)(width - 1)), 0.f);
        y0 = std::max(std::min(y0, (float)(height - 1)), 0.f);
        x1 = std::max(std::min(x1, (float)(width - 1)), 0.f);
        y1 = std::max(std::min(y1, (float)(height - 1)), 0.f);

        objects[i].rect.x = x0;
        objects[i].rect.y = y0;
        objects[i].rect.width = x1 - x0;
        objects[i].rect.height = y1 - y0;
    }

    // sort objects by area
    struct
    {
```

```
    bool operator()(const Object& a, const Object& b) const
    {
        return a.rect.area() > b.rect.area();
    }
} objects_area_greater;
std::sort(objects.begin(), objects.end(), objects_area_greater);

return 0;
}
```

Gambar 4.14 Kode untuk Melakukan Inferensi

4.7 Pembahasan

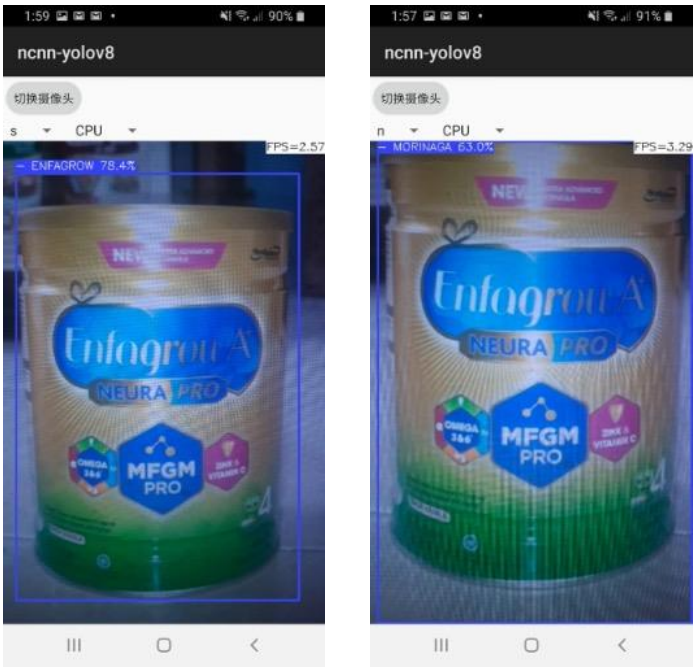
Berdasarkan hasil pelatihan model dan integrasi model yang telah dilakukan, didapatkan hasil untuk masing-masing ukuran dan perangkat bergerak dari YOLOv8 yang dapat dilihat pada Gambar 4.15.

Model	Waktu Inferensi NVIDIA Tesla V100	Waktu Inferensi Samsung A54	Waktu Inferensi Samsung A6	mAP50 PyTorch Model	mAP50 NCNN Model
YOLOV8n	19 ms	41.46 ms	257.73 ms	0.961	0.947
YOLOV8s	19.9 ms	75.19 ms	341.3 ms	0.969	0.962

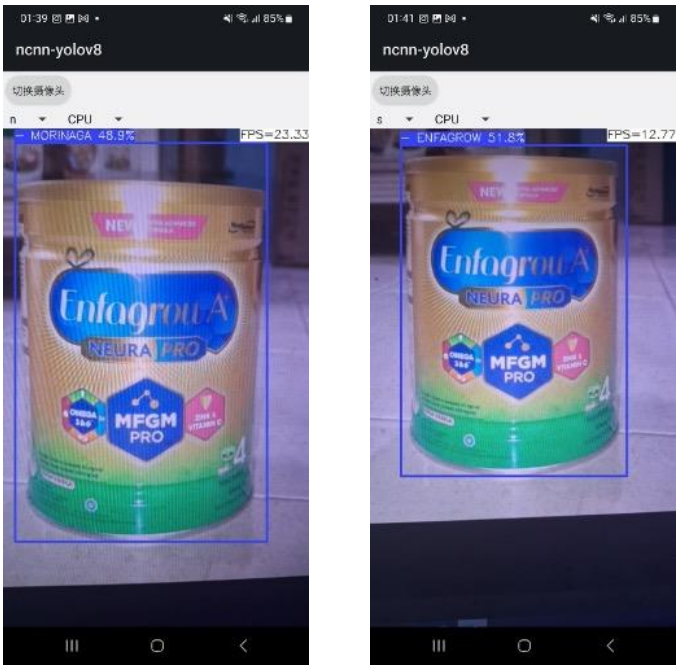
Gambar 4.15 Hasil Evaluasi Berbagai Jenis Perangkat

Berdasarkan hasil tersebut dapat dlihat bahwa mAP pada model YOLOv8s lebih unggul daripada model YOLOv8n, tetapi hanya terpaut sedikit dalam perbandingan model PyTorch maupun NCNN. Namun, saat kita membandingan waktu inferensi terhadap berbagai macam perangkat, dapat dilihat bahwa NVIDIA Tesla V100 dapat menjalankan inferensi dalam waktu 19 ms, sedangkan untuk perangkat bergerak, Samsung A54 jauh lebih unggul daripada Samsung A6, dimana waktu inferensi untuk YOLOv8n dan YOLOv8s untuk Samsung A54 berada di 41.46 ms dan 75.19 ms dan untuk Samsung A6 berada di 257.73 ms dan 341.3 ms. Hal ini dapat terjadi karena kemampuan CPU untuk perangkat Samsung A54 jauh lebih

mumpuni dibanding Samsung A6. Beberapa hasil untuk setiap model dari setiap perangkat bergerak dapat dilihat pada Gambar 4.16 dan Gambar 4.17.



Gambar 4.16 Hasil Inferensi dari Perangkat Samsung A6



Gambar 4.17 Hasil Inferensi Perangkat Samsung A54

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, dapat ditarik beberapa kesimpulan sebagai berikut:

a. Penelitian ini telah berhasil mengimplementasikan YOLOv8 untuk perangkat bergerak untuk melakukan deteksi untuk ketersediaan produk di rak secara *real-time*.
b. Model yang diuji dalam penelitian ini terdiri dari YOLOv8n dan YOLOv8s, dengan hasil deteksi terbaik dihasilkan oleh YOLOv8s dengan mAP50 0.969 untuk model dalam perangkat GPU dan 0.962 untuk model dalam perangkat bergerak.
c. Model YOLOv8n memiliki waktu inferensi paling cepat dibanding dengan ukuran YOLOv8s dengan lama waktu yang dibutuhkan adalah 19 ms untuk perangkat GPU, dibandingakan dengan 19.9 ms, dengan perbedaan hanya 0.09 ms
d. Waktu yang dibutuhkan perangkat bergerak lebih lama daripada perangkat GPU, dengan waktu inferensi untuk Samsung A54 untuk model YOLOv8n dan YOLOv8s berada pada waktu 41.46 ms dan 75.19 ms. Sedangkan untuk Samsung A6 untuk model YOLOv8n dan YOLOv8s berada pada waktu 257.73 ms dan 341.3 ms.
e. Kemampuan komputasi untuk perangkat bergerak Samsung A54 jauh lebih cepat daripada Samsung A6, hal ini dibuktikan dengan hasil inferensi model YOLOv8n dan YOLOv8s.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, terdapat beberapa saran perbaikan yang dapat diimplementasikan pada pengembangan berikutnya. Beberapa saran tersebut adalah sebagai berikut:

a. Meskipun YOLOv8n telah menunjukkan hasil yang baik, penelitian selanjutnya dapat mencoba model lain dengan mengoptimalkan *hyperparameter*. Tujuannya adalah untuk memperbaiki dan meningkatkan performa deteksi secara keseluruhan.
b. Pengujian yang dilakukan pada GPU, Samsung A54, dan Samsung A6 memberikan hasil yang memuaskan. Penelitian selanjutnya dapat mempertimbangkan pengujian pada berbagai jenis perangkat keras untuk memastikan generalitas dan portabilitas model.

c. Penelitian selanjutnya dapat memfokuskan pada implementasi model ini secara langsung dalam aplikasi pengecekan ketersediaan produk.

DAFTAR PUSTAKA

Aastrup, J., & Kotzab, H. (2010). Forty years of Out-of-Stock research – and shelves are still empty. *The International Review of Retail, Distribution and Consumer Research*, *20*(1), 147–164. https://doi.org/10.1080/09593960903498284

Afram, G. (2020). *A study on how to improve On-shelf availability : With deep learning*. Mid Sweden University, Department of Information Systems and Technology.

Amjoud, A. B., & Amrouch, M. (2023). Object Detection Using Deep Learning, CNNs and Vision Transformers: A Review. *IEEE Access*, *11*, 35479–35516. https://doi.org/10.1109/ACCESS.2023.3266093

Ardiansyah, M. N., Muttaqin, P. S., Prasetio, M. D., & Novitasari, N. (2021). Identifikasi Objek/Produk untuk Proses Stock Taking Barang menggunakan Konsep Object Recognition. *Jurnal Rekayasa Sistem & Industri (JRSI)*, *8*(01), 28. https://doi.org/10.25124/jrsi.v8i1.455

Bengio, Y. (2012). Practical Recommendations for Gradient-Based Training of Deep Architectures. In G. B. and M. K.-R. Montavon Grégoire and Orr (Ed.), *Neural Networks: Tricks of the Trade: Second Edition* (pp. 437–478). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-35289-8_26

Campo, K., Gijsbrechts, E., & Nisol, P. (2003). The impact of retailer stockouts on whether, how much, and what to buy. *International Journal of Research in Marketing*, *20*(3), 273–286. https://doi.org/https://doi.org/10.1016/S0167-8116(03)00037-5

Chen, W., Li, Y., Tian, Z., & Zhang, F. (2023). 2D and 3D object detection algorithms from images: A Survey. *Array*, *19*, 100305. https://doi.org/https://doi.org/10.1016/j.array.2023.100305

Cheng, J., Wang, P., Li, G., Hu, Q., & Lu, H. (2018). Recent Advances in Efficient Computation of Deep Convolutional Neural Networks. *CoRR*, *abs/1802.00939*. http://arxiv.org/abs/1802.00939

Chopra, S., & Meindl, P. (2002). Supply Chain Management. Strategy, Planning & Operation. In *Das Summa Summarum des Management* (pp. 265–275). Gabler. https://doi.org/10.1007/978-3-8349-9320-5_22

Corsten, D., & Gruen, T. (2003). Desperately seeking shelf availability: an examination of the extent, the causes, and the efforts to address retail out-of-stocks. *International Journal of*

Retail & Distribution Management, *31*(12), 605–617. https://doi.org/10.1108/09590550310507731

Diwan, T., Anirudh, G., & Tembhurne, J. V. (2023). Object detection using YOLO: challenges, architectural successors, datasets and applications. *Multimedia Tools and Applications*, *82*(6), 9243–9275. https://doi.org/10.1007/s11042-022-13644-y

Doherty, J., Gardiner, B., Kerr, E., Siddique, N., & Manvi, S. S. (2022). *Comparative Study of Activation Functions and Their Impact on the YOLOv5 Object Detection Model* (pp. 40–52). https://doi.org/10.1007/978-3-031-09282-4_4

Dubey, S. R., Singh, S. K., & Chaudhuri, B. B. (2022). *Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark*.

FeiGeChuanShu, & Q-Engineering. (2023). *ncnn-android-yolov8: YOLOv8 implementation on NCNN for Android*.

Fisher, M., Raman, A., & Mcclelland, A. (2000). Rocket Science Retailing Is Almost Here–Are You Ready? *Harvard Business Review*, *78*.

Ganesh, P., Chen, Y., Yang, Y., Chen, D., & Winslett, M. (2021). *YOLO-ReT: Towards High Accuracy Real-time Object Detection on Edge GPUs*.

George, M., & Floerkemeier, C. (2014). *Recognizing Products: A Per-exemplar Multi-label Image Classification Approach* (pp. 440–455). https://doi.org/10.1007/978-3-319-10605-2_29

Goldman, E., Herzig, R., Eisenschtat, A., Ratzon, O., Levi, I., Goldberger, J., & Hassner, T. (2019). *Precise Detection in Densely Packed Scenes*.

Gruen, T. W. (2007). *A Comprehensive Guide To Retail Out-of-Stock Reduction In the Fast-Moving Consumer Goods Industry A research study conducted*. https://api.semanticscholar.org/CorpusID:27530170

Gunawardena, N., Ginige, J. A., Javadi, B., & Lui, G. (2022). Performance Analysis of CNN Models for Mobile Device Eye Tracking with Edge Computing. *Procedia Computer Science*, *207*, 2291–2300. https://doi.org/https://doi.org/10.1016/j.procs.2022.09.288

Hajizadeh, M., Sabokrou, M., & Rahmani, A. (2022). *MobileDenseNet: A new approach to object detection on mobile devices*.

He, K., Girshick, R., & Dollár, P. (2018). *Rethinking ImageNet Pre-training*.

Hübner, A., Amorim, P., Kuhn, H., Minner, S., & Van Woensel, T. (2018). Retail operations. *OR Spectrum*, *40*(4), 831–835. https://doi.org/10.1007/s00291-018-0535-1

Hussain, M. (2024). YOLOv1 to v8: Unveiling Each Variant–A Comprehensive Review of YOLO. *IEEE Access*, *12*, 42816–42833. https://doi.org/10.1109/ACCESS.2024.3378568

Jha, D., Mahjoubfar, A., & Joshi, A. (2022). *Designing an Efficient End-to-end Machine Learning Pipeline for Real-time Empty-shelf Detection*.

Jocher, G. (2020). *YOLOv5 by Ultralytics*. https://doi.org/10.5281/zenodo.3908559

Jocher, G., Chaurasia, A., & Qiu, J. (2023). *Ultralytics YOLO*. https://github.com/ultralytics/ultralytics

Li, K., Zhu, J., & Li, N. (2021). Insect Detection and Counting Based on YOLOv3 Model. *2021 IEEE 4th International Conference on Electronics Technology (ICET)*, 1229–1233. https://doi.org/10.1109/ICET51757.2021.9450898

Li, X., Wang, W., Wu, L., Chen, S., Hu, X., Li, J., Tang, J., & Yang, J. (2020). *Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection*.

Lin Tsung-Yi and Maire, M. and B. S. and H. J. and P. P. and R. D. and D. P. and Z. C. L. (2014). Microsoft COCO: Common Objects in Context. In T. and S. B. and T. T. Fleet David and Pajdla (Ed.), *Computer Vision – ECCV 2014* (pp. 740–755). Springer International Publishing.

Martinez-Alpiste, I., Golcarenarenji, G., Wang, Q., & Alcaraz-Calero, J. M. (2022). Smartphone-based real-time object recognition architecture for portable and constrained systems. *Journal of Real-Time Image Processing*, *19*(1), 103–115. https://doi.org/10.1007/s11554-021-01164-1

Melek, C., Sőnmez, E., & Varlı Albayrak, S. (2019). *Object Detection in Shelf Images with YOLO*. https://doi.org/10.1109/EUROCON.2019.8861817

Nagel, M., Fournarakis, M., Amjad, R. A., Bondarenko, Y., van Baalen, M., & Blankevoort, T. (2021). A White Paper on Neural Network Quantization. *CoRR*, *abs/2106.08295*. https://arxiv.org/abs/2106.08295

Nelson, J., & Solawetz, J. (2020, June 12). *Responding to the Controversy about YOLOv5*. Roboflow Blog. https://blog.roboflow.com/yolov4-versus-yolov5/

Ni, H., & The ncnn contributors. (2017). *ncnn*. https://github.com/Tencent/ncnn

ONNX. (2024). *ONNX/onnx: Open Neural Network Exchange*.

O'Shea, K., & Nash, R. (2015). *An Introduction to Convolutional Neural Networks*.

Padilla, R., Netto, S. L., & da Silva, E. A. B. (2020). A Survey on Performance Metrics for Object-Detection Algorithms. *2020 International Conference on Systems, Signals and*

Image Processing (IWSSIP), 237–242. https://doi.org/10.1109/IWSSIP48289.2020.9145130

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You Only Look Once: Unified, Real-Time Object Detection*.

Selvam, P., Abraham, J., & Koilraj, S. (2022). *A Deep Learning Framework for Grocery Product Detection and Recognition*. https://doi.org/10.21203/rs.3.rs-1431986/v1

Sinha, A., Banerjee, S., & Chattopadhyay, P. (2022). *An Improved Deep Learning Approach For Product Recognition on Racks in Retail Stores*.

Solawetz, J., & Francesco. (2023). *What is YOLOv8? The Ultimate Guide*. https://blog.roboflow.com/whats-new-in-yolov8/

Song, M., & Guo, Q. (2022). Efficient 3D object recognition in mobile edge environment. *Journal of Cloud Computing*, *11*(1), 92. https://doi.org/10.1186/s13677-022-00359-6

Spielmaker, K., Spielmaker, K. J., & Walton, S. M. (2012). *On Shelf Availability: A Literature Review & Conceptual Framework On Shelf Availability: A Literature Review & Conceptual Framework On-Shelf Availability in Retailing: A Literature Review and Conceptual Model On Shelf Availability in Retailing: A Literature Review and Conceptual Framework*. https://scholarworks.uark.edu/mktguht

Sriram, T., Vishwanatha Rao, K., Biswas, S., & Ahmed, B. (1996). Applications of barcode technology in automated storage and retrieval systems. *Proceedings of the 1996 IEEE IECON. 22nd International Conference on Industrial Electronics, Control, and Instrumentation*, *1*, 641–646 vol.1. https://doi.org/10.1109/IECON.1996.571035

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). *Going Deeper with Convolutions*.

Tonioni, A., & Di Stefano, L. (2017). *Product Recognition in Store Shelves as a Sub-Graph Isomorphism Problem* (pp. 682–693). https://doi.org/10.1007/978-3-319-68560-1_61

Varghese, R., & M, S. (2024). YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness. *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, 1–6. https://doi.org/10.1109/ADICS58448.2024.10533619

Vasconcelos, C., Birodkar, V., & Dumoulin, V. (2022). Proper Reuse of Image Classification Features Improves Object Detection. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 13618–13627. https://doi.org/10.1109/CVPR52688.2022.01326

Wang, P., Zinn, W., & Croxton, K. L. (2010). Sizing Inventory When Lead Time and Demand are Correlated. *Production and Operations Management*, *19*(4), 480–484. https://doi.org/https://doi.org/10.1111/j.1937-5956.2009.01109.x

Wu, T., & Dong, Y. (2023). YOLO-SE: Improved YOLOv8 for Remote Sensing Object Detection and Recognition. *Applied Sciences*, *13*(24), 12977. https://doi.org/10.3390/app132412977

Xu, X., Ding, Y., Hu, S., Niemier, M., Cong, J., Hu, Y., & Shi, Y. (2018). Scaling for edge inference of deep neural networks. *Nature Electronics*, *1*. https://doi.org/10.1038/s41928-018-0059-3

Yi, W., Sun, Y., Ding, T., & He, S. (2019). *Detecting retail products in situ using CNN without human effort labeling*.

Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2020). Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(07), 12993–13000. https://doi.org/10.1609/aaai.v34i07.6999

Zhu, X., Vondrick, C., Fowlkes, C. C., & Ramanan, D. (2015). Do We Need More Training Data? *CoRR*, *abs/1503.01508*. http://arxiv.org/abs/1503.01508

Zinn, W., & Liu, P. C. (2008). A COMPARISON OF ACTUAL AND INTENDED CONSUMER BEHAVIOR IN RESPONSE TO RETAIL STOCKOUTS. *Journal of Business Logistics*, *29*(2), 141–159. https://doi.org/10.1002/j.2158-1592.2008.tb00090.x