E-ISSN: 2715-2731

Pengembangan Arsitektur Model YOLOv8 untuk Meningkatkan Performa *Object Detection* pada Varian Boks *Warehouse Palletizing*

Muhammad Alif Amri Muzammil[1,*], Rarasmaya Indraswari[2]

Departemen Sistem Informasi, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember, Indonesia

[1]6026231025@student.its.ac.id; [2]raras@its.ac.id

*penulis korespondensi

INFO ARTIKEL

Sejarah Artikel
Diterima: 5 Juli 2024
Direvisi: 30 Juli 2024
Diterbitkan: 30 Agustus 2024

Kata Kunci
Feature Pyramid Network
Identifikasi Varian
Object Detection
TensorRT
YOLOv8

ABSTRAK

Dalam lingkungan industri modern, paletisasi gudang memainkan peran penting dalam mengoptimalkan rantai pasokan untuk memenuhi permintaan konsumen yang terus meningkat. Namun, banyak perusahaan masih menggunakan proses manual untuk pengaturan palet, yang menyebabkan kesalahan manusia dalam identifikasi dan penempatan barang. Hal ini mengakibatkan inefisiensi operasional dan kerugian material. Di era digital, *computer vision*, terutama *object detection*, sangat penting dalam mengatasi berbagai tantangan industri. Memanfaatkan teknologi pendeteksian objek yang canggih seperti YOLOv8 dapat secara signifikan meminimalkan kesalahan manusia dengan mengaktifkan deteksi barang secara otomatis, sehingga mengurangi keterlibatan manusia. Mengingat data yang digunakan untuk deteksi palet memiliki ukuran yang seragam, efisiensi model YOLOv8 dapat ditingkatkan dengan memodifikasi arsitekturnya ke tingkat multiskala. Penerapan *Feature Pyramid Network* (FPN) yang dimodifikasi dalam arsitektur YOLOv8 meningkatkan efisiensi pelatihan model dengan berfokus pada fitur data yang penting. Penggunaan TensorRT dalamm proses inferensi YOLOv8 untuk meningkatkan kecepatan dan kinerja, sehingga cocok untuk aplikasi *real-time*. Penelitian ini menawarkan solusi komprehensif untuk meningkatkan efisiensi dan akurasi operasional dalam pembuatan palet gudang sekaligus mengurangi kesalahan manusia. Dengan penggunaan model modifikasi arsitektur FPN untuk objek berskala kecil dan implementasi TensorRT memberikan performa terbaik yaitu mAP 95,6% dan performa *inference speed* tercepat yaitu 12ms.

PENDAHULUAN

Paletisasi gudang (*warehouse palletizing*) adalah proses dalam logistik dan manajemen rantai pasokan yang melibatkan pengaturan dan pengorganisasian produk atau barang di atas palet secara sistematis. Dalam lingkungan industri modern, paletisasi gudang merupakan bagian penting dari rantai pasokan kontemporer karena membantu mengatur barang untuk memenuhi permintaan pelanggan yang terus meningkat. Paletisasi dapat meningkatkan efisiensi dan keamanan penanganan dan pengangkutan barang, karena palet dapat dengan mudah dipindahkan dengan forklift atau peralatan lainnya [1]. Industri menghadapi tekanan untuk meningkatkan kecepatan pemenuhan pesanan karena ekspektasi pelanggan untuk pengiriman yang lebih cepat, dengan tren digitalisasi, otomatisasi, dan peningkatan penggunaan Teknologi Informasi dan Komunikasi yang dibayangkan sebagai konsep utama Industri 4.0 [2]. Terlepas dari perkembangan teknologi yang pesat, banyak pabrik yang masih melakukan proses pembuatan palet secara manual, dan kesalahan sering terjadi karena kelalaian manusia. Mengandalkan manusia untuk memverifikasi dan mengonfirmasi detail

barang rentan terhadap kesalahan atau kelalaian [3]. Di era digital ini, teknologi *object detection* telah menjadi perbincangan utama dalam mengatasi berbagai masalah di berbagai industri. *Computer vision* memainkan peran penting dalam Industri 4.0 dengan memungkinkan mesin untuk memahami, menganalisis, dan mengontrol proses produksi [4]. Dalam konteks paletisasi gudang, kemampuan untuk mengidentifikasi dan menempatkan barang secara akurat adalah kunci dalam meningkatkan efisiensi. Pada PT. XYZ, penyusunan paletisasi masih dilakukan secara manual dengan tenaga manusia, sehingga sering terjadi kelalaian atau *human error*. Kelalaian yang terjadi pada PT. XYZ adalah penyusunan varian boks yang salah pada susunan palet yang telah ditentukan. Karena kesalahan penyusunan inilah perusahaan mengalami kerugian disebabkan oleh varian produk yang berbeda memiliki harga yang berbeda juga. Oleh karena itu diperlukan solusi komprehensif dengan membuat sebuah sistem pengecekan susunan palet menggunakan deteksi objek sehingga dapat membantu mengurangi kerugian perusahaan dan meningkatkan efisiensi dari paletisasi gudang pada PT. XYZ.

Deep learning telah digunakan untuk pengenalan kotak logistik dalam prosedur paletisasi robot industri [5]. Banyak algoritma yang telah dikembangkan untuk mengatasi tantangan tersebut. Salah satu penelitian oleh Yoon dkk. menggunakan *deep learning* untuk melakukan pengenalan kotak logistik dalam prosedur paletisasi robot industri. Terdapat tantangan dalam mengenali dan memilih kotak dalam sistem pemindahan palet otomatis yang digunakan dalam logistik. Kerumitan muncul dari susunan kotak yang berantakan dan permukaan luarnya yang bervariasi, yang sering kali ditutupi oleh label, label, dan simbol. Untuk mengatasi masalah tersebut, diusulkan pendekatan deteksi baru menggunakan model berbasis Mask R-CNN, didukung oleh Cycle-GAN. Cycle-GAN mengoptimalkan permukaan luar kotak dengan menghapus tag dan simbol asing sebelum terdeteksi. Metode yang diusulkan secara signifikan meningkatkan kinerja deteksi kotak dalam sistem paletisasi robotik [6]. Pada penelitian ini memberikan wawasan bahwa penggunaan deteksi objek untuk mendeteksi boks pada susunan palet dapat memberikan bantuan secara otomatis kepada pekerja sehingga dapat meminimalkan kesalahan saat melakukan penyusunan palet.

Penelitian lain oleh Zendehdel dkk. mengembangkan sebuah pendeteksi langsung untuk berbagai macam alat yang ada pada *Smart Manufacturing* menggunakan YOLOv5. Masalah yang sering terjadi sebelumnya adalah akurasi dan performa secara *real-time* dalam sistem manufaktur perusahaan. Metode yang disarankan menggunakan model YOLOv5 yang disempurnakan dengan menemukan rata-rata akurasi sebesar 98,3% menunjukkan presisi tinggi dalam alat deteksi *real-time* [4]. Penelitian lain dilakukan oleh L. Jiang dkk. dengan melakukan penyempurnaan model YOLOv5 untuk deteksi rambu lalu lintas yang lebih baik, dengan fokus pada penggabungan dan ektraksi fitur. Pada penelitian ini, FPN (*Feature Pyramid Network*) dimanfaatkan untuk meningkatkan deteksi rambu lalu lintas, terutama yang berukuran kecil [7]. Membuat *Balanced Feature Pyramid* dimana struktur ini menggabungkan fitur pada skala berbeda, menyeimbangkan kekayaan semantik dan resolusi spasial, yang sangat penting untuk mendeteksi objek yang lebih kecil [8]. Meningkatkan *feature fusion* dengan menggabungkan fitur tingkat rendah dan fitur tingkat tinggi, sehingga meningkatkan performa deteksi [9]. Dari penelitian tersebut, dapat dilihat bahwa modifikasi FPN telah dilakukan pada YOLOv5 dan memberikan peningkatan performa deteksi.

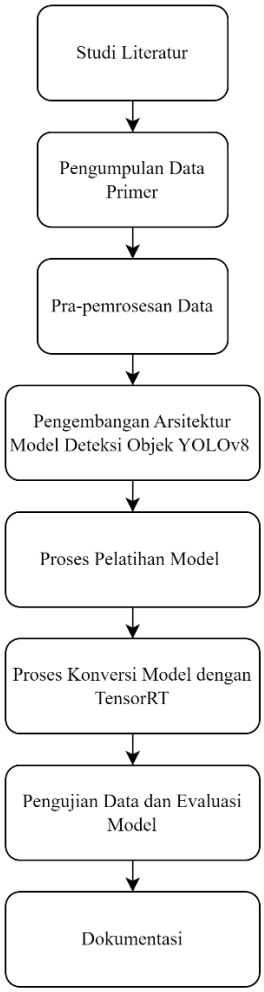
Terdapat banyak algoritma *object detection*, dan terdapat kemungkinan algoritma tersebut dapat digunakan dalam identifikasi pembuatan palet di gudang. Salah satu algoritma yang menarik adalah YOLO (*You Only Look Once*). Perkembangan YOLO cukup pesat dalam dunia computer vision. YOLO merupakan salah satu pendekatan yang melibatkan pemanfaatan kekuatan dari *deep learning* [10]. Perkembangan YOLO yang terbaru adalah YOLOv8, model *computer vision* terbaru yang dikembangkan oleh Ultralytics pada tanggal 10 Januari 2023 [11]. Salah satu tantangan dari model YOLOv8 adalah *inference speed* yang

lebih lambat jika dibandingkan dengan YOLOv5 [12]. Meskipun YOLOv8 memiliki akurasi yang lebih baik dibandingkan YOLOv5, namun kecepatan dalam mendeteksi objek pada citra yang dimiliki YOLOv8 lebih lama jika dibandingkan dengan YOLOv5. Oleh karena itu, beberapa penelitian telah menggunakan berbagai metode yang dapat meningkatkan performa dari model tersebut. Modifikasi *Feature Pyramid Network* (FPN) dilakukan untuk meningkatkan kecepatan arsitektur model YOLOv8 sehingga dapat meningkatkan efisiensi dari arsitektur model tersebut [13]. Penelitian lain mengusulkan TensorRT untuk mengoptimalkan kinerja dengan meningkatkan *inference speed* model *deep learning*, sehingga cocok untuk aplikasi *real-time* [14]. TensorRT secara signifikan meningkatkan efisiensi model *deep learning* untuk *inference real-time*, yang sangat penting untuk sistem tertanam dan aplikasi dengan persyaratan latensi yang ketat [15]. Bellou dan Pisica dkk. berhasil mengoptimalkan model YOLOv8 dengan mengimplementasikan TensorRT, meningkatkan kinerja inferensi [16]. Teknik optimasi pada TensorRT, seperti kuantisasi dan *layer fusion*, secara nyata dapat meningkatkan kecepatan model, selaras dengan pesyaratan perangkat keras. Hasil percobaan menunjukkan bahwa TensorRT dapat meningkatkan *inference speed* dan mencapai kriteria kecepatan untuk aplikasi *real-time* [16]. Dalam hal ini, modifikasi FPN diharapkan dapat melakukan pelatihan lebih cepat dengan tetap memiliki *inference speed* yang lebih singkat untuk memenuhi kebutuhan sistem yang akan digunakan secara *real-time*. Meskipun telah banyak penelitian mengenai berbagai versi YOLO, namun belum penelitian yang menggunakan YOLOv8 jumlahnya terbilang cukup sedikit.

Oleh karena itu, penelitian ini mengusulkan model arsitektur YOLOv8 dengan modifikasi FPN pada arsitektur untuk meningkatkan efisiensi pelatihan dengan kumpulan data *warehouse palletizing*. Selain itu, TensorRT digunakan untuk meningkatkan performa *real-time* dengan meningkatkan *inference speed* dari model. Sehingga perlu dilakukan eksperimen untuk dapat meningkatkan *inference speed* tanpa mengorbankan akurasi dari model dan layak untuk pengaplikasian secara *real-time*. Kontribusi yang diberikan pada penelitian ini adalah mengetahui model yang cocok digunakan dalam meningkatkan kecepatan tanpa mengurangi akurasi secara signifikan pada studi kasus PT.XYZ ini, dimana motode yang memberikan akurasi terbaik adalah model *Small* dengan impelementasi TensorRT YOLOv8 dengan mAP 95,6% dan *inference speed* 12ms.

METODE

Penelitian ini dilakukan sesuai dengan alur penelitian yang telah divisualisasikan pada Gambar 1. Terdapat beberapa tahapan yang dilakukan pada penelitian ini. Pertama. Studi literatur untuk mendukung pengerjaan penelitian sebagai acuan dalam melakukan penelitian. Kedua, pengumpulan data primer. Ketiga, pra-pemrosesan data, dengan melakukan anotasi atau pemberian label pada data primer yang masih *raw* atau mentah menjadi dataset dengan format YOLOv8, sehingga data dapat digunakan untuk melakukan pelatihan model. Keempat, pengembangan arsitektur model deteksi objek YOLOv8 dengan tujuan meningkatkan efisiensi pembuatan atau pelatihan model. Kelima, proses pelatihan model dengan beberapa modifikasi arsitektur yang telah dilakukan pada model. Keenam, proses konversi model dengan TensorRT, dengan tujuan meningkatkan performa dari model yang telah dilatih. Ketujuh, pengujian data dan evaluasi model, untuk menguji model mana yang memiliki performa paling optimal dan evaluasi model. Terakhir, dokumentasi yang berupa penulisan penelitian.



Gambar 1. Diagram Metode Penelitian

Data

Data yang digunakan merupakan data primer berupa gambar susunan kotak palet pada PT. XYZ. Terdapat dua varian produk dan masing-masing varian memiliki dua pola yang berbeda. *Dataset* berisi jenis varian kotak 60ml dan 63ml.

Gambar 2. Contoh Citra pada *Dataset Warehouse Palletizing*

Jumlah total data adalah 400 gambar. Setiap varian memiliki dua pola, setiap pola memiliki contoh gambar dengan susunan yang benar dan salah, setiap susunan benar atau

salah memiliki 50 gambar. Hal ini dilakukan karena *layer* paling bawah telah diperiksa pada proses sebelumnya. Semuanya diperiksa per-*layer* dari *layer* pertama hingga *layer* terakhir, sehingga tidak ada *layer* yang salah berupa kekurangan kotak ataupun terdapat varian yang menempati tempat yang salah. Contoh citra pada sebuah *layer* dapat dilihat pada Gambar 2. Contoh Citra pada *Dataset Warehouse Palletizing* Data citra yang diperoleh merupakan data primer dengan citra berwarna yang diambil menggunakan kamera *smartphone* Xiaomi Redmi Note 5 dengan spesifikasi pengambilan citra pada Tabel 1. Proses paletisasi dilakukan per-*layer*, sehingga hanya *layer* bagian atas saja yang diperiksa tanpa melihat *layer* di bawahnya.

Tabel 1. Spesifikasi Pengambilan Citra

Property	***Value***
Dimensions	4000 x 2000
Width	4000 pixels
Height	2000 pixels
Horizontal resolution	72 dpi
Vertical resolution	72 dpi
Bit depth	24
Resolution unit	2
Color representation	sRGB
F-stop	f/1.9
Exposure time	1/20 sec
ISO speed	ISO-640
Focal length	4 mm
Metering mode	*Center Weighted Average*
Flash mode	*No flash, compulsory*
35mm focal length	24
With balance	*Auto*
EXIF version	0220
Item type	JPG

Prapemrosesan Data

Prapemrosesan data adalah langkah mendasar dalam *machine learning*, terutama untuk *deep learning*, yang melibatkan transformasi *raw data* (data mentah atau awal) ke dalam format data yang dapat diproses oleh *neural networks* secara efektif [17]. Pada kasus ini prapemrosesan data yang dilakukan adalah anotasi objek sehingga data citra yang digunakan dapat menjadi input untuk model YOLOv8. Proses anotasi objek dilakukan pada gambar yang telah dikumpulkan selama pengumpulan data primer. Anotasi objek dilakukan secara manual kepada 400 citra data primer. Label Studio digunakan untuk memberikan anotasi objek pada kumpulan data gambar. Terdapat dua kelas dalam anotasi data, yaitu varian 60ml dan 63ml. Setiap citra memiliki 12 objek boks yang perlu dilakukan anotasi, sehingga total objek yang dilakukan anotasi adalah 4800 objek menggunakan aplikasi Label Studio.

Pada Gambar 3 dapat dilihat tampilan dari *tools* Label Studio saat proses anotasi dilakukan. Dapat dilihat bahwa pada contoh citra yang digunakan terdapat 4 kotak berwarna kuning dan 8 kotak berwarna hitam, kotak tersebut biasa disebut *bounding box*. Kotak berwarna kuning menandakan bahwa objek yang dianotasi merupakan produk 63ml, dan kotak berwarna hitam menandakan bahwa objek yang dianotasi merupakan produk 60ml. Setiap *layer* harus sudah dipastikan berisi 12 kotak, dan keseluruhan kotak merupakan produk yang sama.

Gambar 3. Contoh Anotasi Citra *Dataset Warehouse Palletizing*

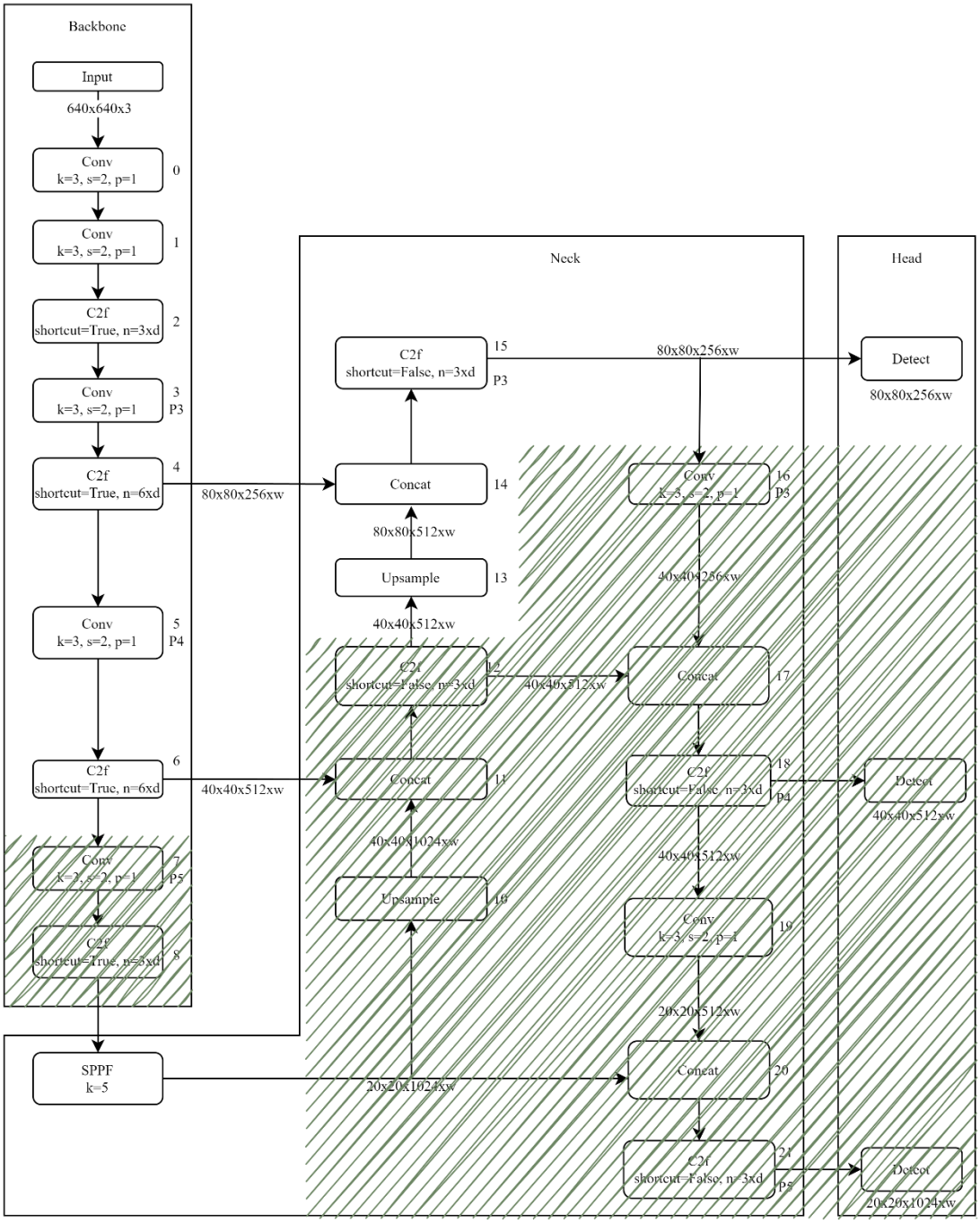
Modifikasi Arsitektur YOLOv8

YOLOv8 adalah versi terbaru dari keluarga model *object detection* "*You Only Look Once*" (YOLO). YOLO dikenal dengan kecepatan dan akurasinya yang tinggi, sehingga menjadi pilihan populer untuk tugas *object detection* secara *real-time*. YOLOv8 telah dimodifikasi dalam pengembangan penelitian ini dengan melompati beberapa *layer* pada arsitektur. Modifikasi ini bertujuan untuk mempercepat proses pelatihan sesuai dengan data yang digunakan, dalam hal ini dataset yang digunakan memiliki objek dengan ukuran skala yang sama.

Gambar 4 menunjukkan YOLOv8 yang telah dimodifikasi pada *Feature Pyramid Network*, terlihat bahwa bagian yang diarsir akan dilewati sehingga model hanya akan menggunakan satu *detection head*, dalam contoh ini adalah *detection head* untuk objek-objek berskala kecil pada sebuah gambar. Pada penelitian ini, dibuat modifikasi YOLOv8 yang akan menggunakan masing-masing *detection head* secara mandiri, sehingga akan ada tiga modifikasi arsitektur yang bergantung pada ukuran objek: yaitu kecil, sedang, dan besar. Pada Gambar 4 dapat dilihat bahwa hanya P3 yang digunakan untuk melakukan training, P4 dan P5 merupakan *detection head* untuk skala ukuran sedang dan besar. Sehingga input citra yang dilatih langsung menuju *Upsample* untuk mengembalikan ukuran menjadi 80x80.

YOLO sendiri memiliki tiga komponen utama yang memiliki peran penting dalam proses *object detection*: *Backbone*, *Neck*, dan *Head*. Pada Gambar 4 dapat diliihat bagian *backbone*, *neck*, dan *head* dibatasi dengan kotak masing-masing. Ketiga komponen ini memiliki peran masing-masing dan saling berhubungan. Pertama *Backbone*, berfungsi sebagai langkah untuk melakukan ekstraksi fitur (*feature extraction*) dari gambar *input*. Ekstraksi fitur merupakan proses yang dilakukan untuk mengubah data mentah menjadi fitur numerik yang dapat diproses dengan tetap mempertahankan informasi dalam kumpulan data asli [18]. Proses ini memungkinkan pengurangan dimensi data, sehingga lebih mudah dikelola oleh algoritma *machine learning*. Tujuannya adalah untuk fokus pada detail penting dengan mengabaikan informasi yang kurang penting. *Backbone* merupakan jaringan *Convolutional Neural Network* (CNN) mendalam yang mengambil gambar input dan menghasilkan peta fitur (*feature maps*).

Ketiga *Head*, merupakan bagian yang bertanggung jawab untuk melakukan deteksi dan klasifikasi objek berdasarkan peta fitur yang dihasilkan oleh *Neck*. Bagian ini berfungsi untuk melakukan prediksi kotak pembatas *(bounding boxes)*, skor kepercayaan *(confidence scores)*, dan klasifikasi objek. Setiap sel dalam *grid* menghasilkan beberapa *bounding boxes*, setiap *bounding boxes* memiliki koordinat, *confidence score*, dan probabilitas kelas. Peta fitur dari *Neck* melewati beberapa lapisan tambahan untuk mengeluarkan prediksi akhir sehingga menghasilkan prediksi *bounding boxes, confidence scores*, dan probabilitas kelas.



Gambar 4. Arsitektur YOLOv8 dengan Modifikasi FPN untuk Objek Skala Kecil

Kedua *Neck*, adalah komponen yang menghubungkan *Backbone* dengan *Head*. Bertanggung jawab sebagai pengolah peta fitur yang dihasilkan oleh backbone agar lebih berguna untuk *object detection*. Peta fitur adalah representasi dari fitur-fitur yang telah dipelajari dan diekstraksi dari gambar *input* oleh CNN [19]. Struktur *Neck* terdiri dari beberapa lapisan *convolutional* tambahan, serta teknik seperti FPN yang membantu dalam menghasilkan peta fitur multi-skala, yang penting untuk mendeteksi objek dengan berbagai ukuran. Pada penelitian ini, bagian FPN inilah yang dimodifikasi sehingga menyesuaikan dengan ukuran objek dari kumpulan data *warehouse palletizing.*

Dengan menggunakan arsitektur ini, pelatihan yang dilakukan dapat lebih efisien karena Segmentasimemfokuskan hanya kepada satu skala ukuran objek pada data citra. Dengan

melakukan lompatan terhadap beberapa *layer*, waktu pelatihan yang dibutuhkan menjadi lebih singkat. Hal ini dapat mengurangi biaya sumber daya yang dibutuhkan oleh perusahaan. Informasi yang didapat oleh model juga lebih maksimal karena langsung menyesuaikan *feature* yang digunakan sesuai dengan karakteristik objek pada kumpulan data. Selain itu, *inference speed* yang dimiliki model menjadi lebih optimal dengan melakukan pemangkasan arsitektur.

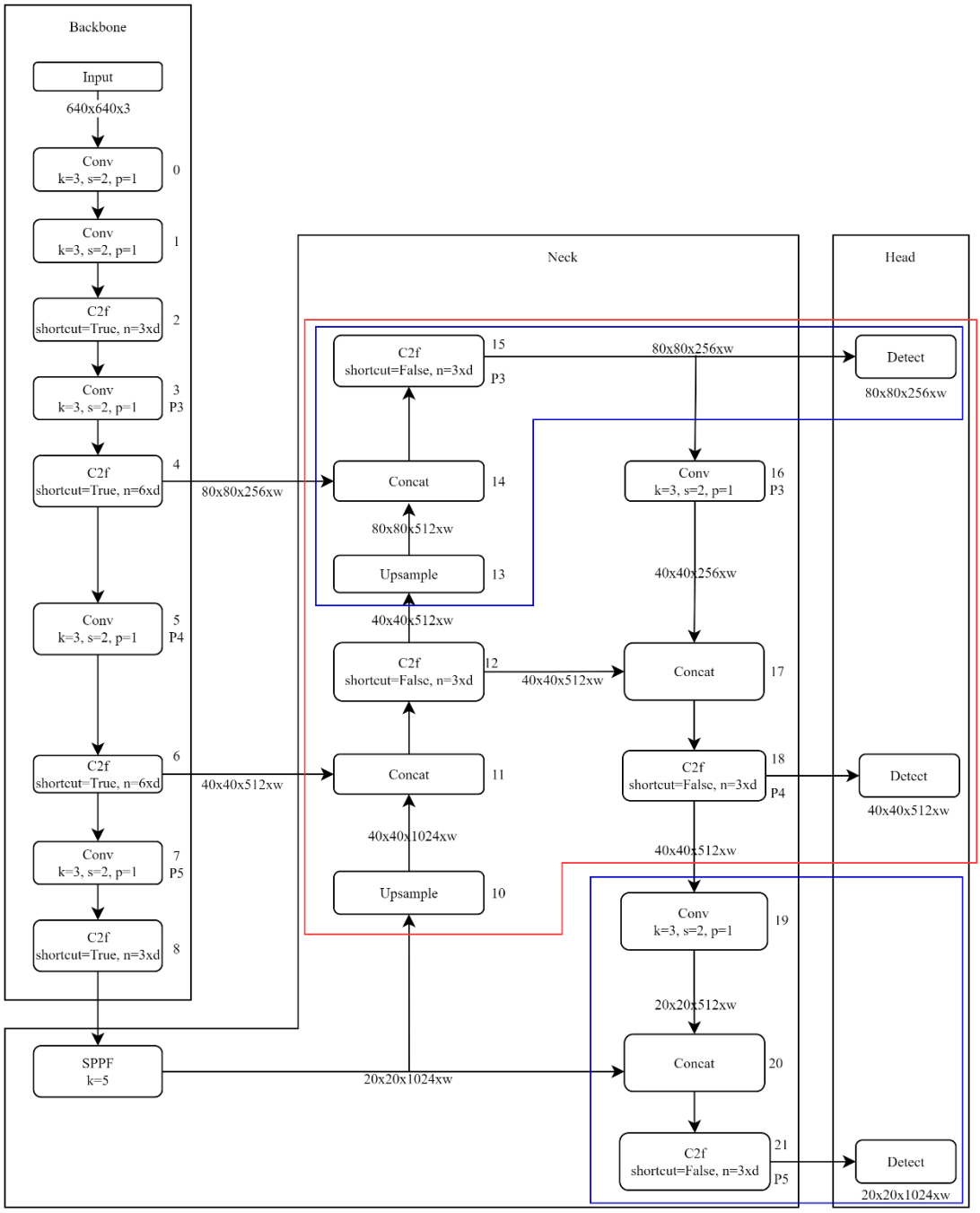
Metrik Evaluasi

Terdapat dua metrik evaluasi yang digunakan untuk menilai peningkatan performa dari modifikasi arsitektur yang diusulkan: yaitu mAP dan *inference speed*. mAP (*mean Average Precision*) adalah metrik yang digunakan untuk mengevaluasi performa dari model *object detection*. mAP dihitung dengan mengambil nilai presisi rata-rata atau *Average Precision* (AP) untuk semua kelas dan menghitung rata-ratanya. Hal ini memberikan skor efektifitas keseluruhan untuk kemampuan model dalam mengidentifikasi dan menemukan lokasi objek dengan benar di berbagai kategori [20]. Terdapat dua jenis mAP pada YOLOv8 yang dibedakan dari letak pada rentang ambang batas Intersection over Union (IoU), yaitu mAP50 dan mAP50-95. Pada dasarnya mAP50-95 merupakan metrik yang lebih ketat dan terperinci karena memperhitungkan ketepatan model di seluruh spektrum ambang batas IoU, bukan hanya satu ambang batas seperti mAP50. Sedangkan *Inference speed*, mengacu kepada waktu yang dibutuhkan model terlatih untuk memproses input dan menghasilkan output [21]. Mengoptimalkan *inference speed* sangat penting untuk aplikasi *real-time*.

HASIL DAN PEMBAHASAN

Pada penelitian ini, Colab Pro digunakan sebagai *notebook* untuk menjalankan pelatihan dan pengujian arsitektur YOLOv8. Spesifikasi GPU yang digunakan pada Colab Pro adalah T4 dengan RAM 32GB. Berbagai metrik evaluasi digunakan untuk menilai kinerja model yang diusulkan. Untuk mengukur akurasi *object detection*, mAP (*mean Average Precision*) digunakan untuk mengevaluasi kinerja model *object detection*. mAP dihitung dengan mengambil nilai rata-rata presisi (AP) untuk semua kelas dan merata-ratakannya. Hal ini memberikan nilai efektivitas keseluruhan untuk kemampuan model untuk mengidentifikasi dan menemukan objek dalam berbagai kategori dengan benar. Untuk mengukur kemampuan model yang akan digunakan pada aplikasi *real-time*, digunakan *inference speed* sebagai metrik untuk dapat mengevaluasi apakah model YOLOv8 yang telah dilatih dapat diimplementasikan pada aplikasi secara *real-time*. *Inference speed* merupakan metrik yang sangat penting terutama pada aplikasi *real-time* yang membutuhkan respon cepat. Konfigurasi yang digunakan sama pada semua skenario eksperimen, yaitu menggunakan workers = 2, batch = 12, devices = 0, epochs = 100, dan patience = 50.

Dalam penelitian ini terdapat empat eksperimen yang telah dilakukan, yaitu menggunakan arsitektur YOLOv8 orisinil, dan arsitektur yang telah dimodifikasi FPN dengan masing-masing modifikasi berupa *object detection* khusus untuk skala objek kecil, sedang, dan besar. Pada eksperimen ini, *input size* yang digunakan adalah 640x640. Pada Gambar 5 menunjukkan arsitektur YOLOv8 yang telah dimodifikasi untuk objek skala sedang dan besar. Untuk modifikasi arsitektur model skala sedang, *layer* yang berada dalam kotak biru tidak digunakan. Sedangkan untuk modifikasi arsitektur model skala besar, *layer* yang berada di dalam kotak merah tidak digunakan atau dilompati. Mengurangi penggunaan *layer* dari total keseluruhan pada arsitektur 23 lapisan sebelum citra terdeteksi.



Gambar 5. Arsitektur YOLOv8 dengan Modifikasi FPN untuk Objek Skala Sedang dan Besar

Tabel 2 menunjukkan hasil pelatihan keempat skenario model dengan menggunakan *input size* 640x640. Dapat dilihat bahwa model yang diusulkan, yaitu model modifikasi arsitektur FPN untuk objek berskala kecil, memiliki performa paling maksimal diantara tiga model yang lain. Berdasarkan pelatihan model, model *Small* memiliki kecepatan waktu pelatihan paling singkat, yaitu 0,493 jam; dan memberikan nilai mAP50-95 yang paling tinggi, yaitu 95,5%. Begitu pula saat dilakukan pengujian, nilai mAP50-95 dari model *Small* lebih tinggi apabila dibandingkan dengan tiga model yang lain, yaitu 95,5%. *Inference speed* yang dimiliki model *Small* juga paling cepat, yaitu 19,1ms. Berdasarkan eksperimen tersebut, menandakan bahwa model *one-scale object detection* yang paling sesuai untuk

kumpulan data *warehouse palletizing* PT. XYZ adalah model modifikasi arsitektur FPN untuk objek berskala kecil, atau mode *Small*.

Tabel 2. Hasil Eksperimen Modifikasi Arsitektur Input Size 640x640

Model	***Training***		***Test***	***Predict Inference (ms)***	***Inference Speed (ms)***
	Kecepatan (jam)	**mAP50-95 (%)**	**mAP50-95 (%)**		
Original	0,681	95,2	95,4	39,5	23,4
Small	**0,493**	**95,5**	**95,5**	**27,5**	**19,1**
Medium	0,650	94,5	94,6	38,2	20,9
Large	0,752	90,4	90,6	40,3	20,2

Pada ukuran skala yang lain, skala sedang dan besar, tidak terdapat peningkatan nilai mAP pada saat dilakukan pelatihan. Model *Medium* memiliki kecepatan pelatihan yang lebih singkat, namun menghasilkan nilai mAP yang tidak lebih tinggi dibandingkan model *Original.* Sedangkan model *Large*, memiliki kecapatan yang lebih lambat yaitu 0,752 jam, dan nilai mAP yang dihasilkan juga tidak lebih bagus dibandingkan model *Original.* Meskipun demikian, *inference speed* yang dimiliki kedua model tersebut lebih cepat apabila dibandingkan dengan model *Original.* Model *Medium* dengan *inference speed* 20,9ms dan model *Large* dengan *inference speed* 20,2ms berhasil lebih cepat dibandingkan model *Original* yang memiliki *inference speed* 23,4ms.

Tabel 3. Hasil Eksperimen Implementasi TensorRT pada Modifikasi Arsitektur

Model	***Test***	***Predict Inference (ms)***	***Inference Speed (ms)***
	mAP50-95 (%)		
Small	**95,6**	**12,3**	**12,0**
Medium	95	15,2	12,5
Large	90,8	14,4	16,2

Eksperimen dengan mengimplementasikan TensorRT pada model arsitektur yang telah dimodifikasi juga telah dilakukan. Hasil eksperimen implementasi TensorRT pada model dengan modifikasi arsitektur dapat dilihat pada Tabel 3. Dapat dilihat bahwa penggunaan TensorRT memiliki pengaruh yang signifikan pada *inference speed* dengan peningkatan sebesar 7,1ms lebih cepat apabila dibandingkan dengan model tanpa implementasi TensorRT. Skor mAP50-95 yang dihasilkan juga mengalami peningkatan sebesar 0,1%, dari yang sebelumnya 95,5% menjadi 95,6%.

KESIMPULAN

Pada penelitian ini, telah dikembangkan sebuah modifikasi arsitektur pada model *object detection* YOLOv8 berdasarkan skala objek pada citra yang digunakan. Terdapat tiga modifikasi arsitektur yang dibuat, yaitu model untuk objek dengan skala kecil, sedang, dan besar. Model *object detection* yang diusulkan telah dilatih dengan *input* berupa kumpulan data *warehouse palletizing* pada PT. XYZ dengan *input size* 640x640. Modifikasi arsitektur YOLOv8 untuk *Feature Pyramid Network* (FPN) skala kecil atau model *Small* yang diusulkan memiliki hasil yang memuaskan dengan peningkatan kecepatan pelatihan (*training*) sebesar 1,38 kali pada kumpulan data *warehouse palletizing* dengan 100 *epochs*. Model *Small* menghasilkan peningkatan kecepatan inferensi sebesar 4,3ms dibandingkan dengan model *Original.* Implementasi TensorRT pada tiga model modifikasi; *Small, Medium*, dan *Large* memberikan peningkatan *inference speed* sebesar 1,5 kali dibandingkan

dengan *inference speed* model modifikasi sebelum mengimplementasikan TensorRT. Terdapat peningkatan mAP50-95 sebesar 0,3% pada model *Small* saat pelatihan, jika dibandingkan dengan model *Original*. Selain itu, terdapat peningkatan mAP50-95 pada model *Small* saat dilakukan pengujian (*test*) dengan peningkatan sebesar 0,1% dari model *Original*. Terdapat juga peningkatan mAP50-95 pada model *Small* dengan implementasi TensorRT sebesar 0,1% dari model *Small*, sehingga menjadikan model *Small* dengan implementasi TensorRT memiliki performa paling optimal pada kumpulan data *warehouse palletizing*. Untuk penelitian selanjutnya, dapat dilakukan eksperimen atau penerapan model modifikasi arsitektur FPN pada kumpulan data yang berbeda dan memiliki jumlah citra lebih banyak sehingga dapat digunakan untuk menguji kinerja model.

REFERENSI

[1] T. Borangiu and S. Răileanu, "A smart palletising planning and control model in Logistics 4.0 framework," *Int. J. Prod. Res.*, vol. 61, no. 24, pp. 8580–8597, Dec. 2023, doi: 10.1080/00207543.2022.2154405.

[2] W. S. Alaloul, M. S. Liew, N. A. W. A. Zawawi, and I. B. Kennedy, "Industrial Revolution 4.0 in the construction industry: Challenges and opportunities for stakeholders," *Ain Shams Eng. J.*, vol. 11, no. 1, pp. 225–230, Mar. 2020, doi: 10.1016/J.ASEJ.2019.08.010.

[3] S. Jagtap and S. Rahimifard, "The digitisation of food manufacturing to reduce waste – Case study of a ready meal factory," *Waste Manag.*, vol. 87, pp. 387–397, Mar. 2019, doi: 10.1016/J.WASMAN.2019.02.017.

[4] N. Zendehdel, H. Chen, and M. C. Leu, "Real-time tool detection in smart manufacturing using You-Only-Look-Once (YOLO)v5," *Manuf. Lett.*, vol. 35, pp. 1052–1059, Aug. 2023, doi: 10.1016/J.MFGLET.2023.08.062.

[5] M. Woschank, E. Rauch, and H. Zsifkovits, "A Review of Further Directions for Artificial Intelligence, Machine Learning, and Deep Learning in Smart Logistics," *Sustain. 2020, Vol. 12, Page 3760*, vol. 12, no. 9, p. 3760, May 2020, doi: 10.3390/SU12093760.

[6] J. Yoon, J. Han, and T. P. Nguyen, "Logistics box recognition in robotic industrial de-palletising procedure with systematic RGB-D image processing supported by multiple deep learning methods," *Eng. Appl. Artif. Intell.*, vol. 123, p. 106311, Aug. 2023, doi: 10.1016/j.engappai.2023.106311.

[7] L. Jiang, H. Liu, H. Zhu, and G. Zhang, "Improved YOLO v5 with balanced feature pyramid and attention module for traffic sign detection," *MATEC Web Conf.*, vol. 355, p. 03023, 2022, doi: 10.1051/matecconf/202235503023.

[8] T. Zhang, X. Zhang, J. Shi, S. Wei, J. Wang, and J. Li, "Balanced Feature Pyramid Network for Ship Detection in Synthetic Aperture Radar Images," *IEEE Natl. Radar Conf. - Proc.*, vol. 2020-Septe, Sep. 2020, doi: 10.1109/RadarConf2043947.2020.9266519.

[9] X. Li, D. Song, and Y. Dong, "Hierarchical Feature Fusion Network for Salient Object Detection," *IEEE Trans. Image Process.*, vol. 29, pp. 9165–9175, 2020, doi: 10.1109/TIP.2020.3023774.

[10] G. G. Casas, Z. H. Ismail, M. M. C. Limeira, A. A. L. da Silva, and H. G. Leite, "Automatic Detection and Counting of Stacked Eucalypt Timber Using the YOLOv8 Model," *For. 2023, Vol. 14, Page 2369*, vol. 14, no. 12, p. 2369, Dec. 2023, doi: 10.3390/F14122369.

[11] J. Solawetz and Francesco, "What is YOLOv8? The Ultimate Guide.," *Roboflow*, 2023. https://blog.roboflow.com/whats-new-in-yolov8/ (accessed Jan. 22, 2024).

[12] E. Casas, L. Ramos, E. Bendek, and F. Rivas-Echeverria, "YOLOv5 vs. YOLOv8: Performance Benchmarking in Wildfire and Smoke Detection Scenarios," *J. Image Graph.*, vol. 12, no. 2, pp. 127–136, 2024, doi: 10.18178/joig.12.2.127-136.

[13] Y. Li, S. Zhou, and H. Chen, "Attention-based fusion factor in FPN for object detection," *Appl. Intell.*, vol. 52, no. 13, pp. 15547–15556, 2022, doi: 10.1007/s10489-022-03220-0.

[14] M. Sohan, T. Sai Ram, and C. V. Rami Reddy, "A Review on YOLOv8 and Its Advancements," pp. 529–545, 2024, doi: 10.1007/978-981-99-7962-2_39.

[15] Y. Zhou and K. Yang, "Exploring TensorRT to Improve Real-Time Inference for Deep Learning," *Proc. - 24th IEEE Int. Conf. High Perform. Comput. Commun. 8th IEEE Int. Conf. Data Sci. Syst. 20th IEEE Int. Conf. Smart City 8th IEEE Int. Conf. Dep*, pp. 2011–2018, 2022, doi: 10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00299.

[16] E. Bellou, I. Pisica, and K. Banitsas, "Aerial Inspection of High-Voltage Power Lines Using YOLOv8 Real-Time Object Detector," *Energies 2024, Vol. 17, Page 2535*, vol. 17, no. 11, p. 2535, May 2024, doi: 10.3390/EN17112535.

[17] J. Li, H. Fu, K. Hu, and W. Chen, "Data Preprocessing and Machine Learning Modeling for Rockburst Assessment," *Sustain.*, vol. 15, no. 18, p. 13282, Sep. 2023, doi: 10.3390/SU151813282/S1.

[18] A. Z. da Costa, H. E. H. Figueroa, and J. A. Fracarolli, "Computer vision based detection of external defects on tomatoes using deep learning," *Biosyst. Eng.*, vol. 190, pp. 131–144, Feb. 2020, doi: 10.1016/J.BIOSYSTEMSENG.2019.12.003.

[19] D. Bhatt *et al.*, "CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope," *Electron. 2021, Vol. 10, Page 2470*, vol. 10, no. 20, p. 2470, Oct. 2021, doi: 10.3390/ELECTRONICS10202470.

[20] R. Y. Ju and W. Cai, "Fracture detection in pediatric wrist trauma X-ray images using YOLOv8 algorithm," *Sci. Reports 2023 131*, vol. 13, no. 1, pp. 1–13, Nov. 2023, doi: 10.1038/s41598-023-47460-7.

[21] Z. Liu, T. Zheng, G. Xu, Z. Yang, H. Liu, and D. Cai, "Training-Time-Friendly Network for Real-Time Object Detection," *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 07, pp. 11685–11692, Apr. 2020, doi: 10.1609/AAAI.V34I07.6838.