ANALISIS KINERJA TENSORRT DAN OPENVINO SEBAGAI *INFERENCE FRAMEWORK* UNTUK PEMODELAN DEEP LEARNING

TUGAS AKHIR

LOUIS AXEL TJOKRO ATMODJO
312110008

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI DAN DESAIN
UNIVERSITAS MA CHUNG
MALANG
2024

LEMBAR PENGESAHAN TUGAS AKHIR

ANALISIS KINERJA TENSORRT DAN OPENVINO SEBAGAI *INFERENCE FRAMEWORK* UNTUK PEMODELAN *DEEP LEARNING*

Oleh:

LOUIS AXEL TJOKRO ATMODJO

NIM. 312110008

dari:

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI DAN DESAIN

UNIVERSITAS MA CHUNG

Telah dinyatakan lulus dalam melaksanakan Tugas Akhir sebagai syarat kelulusan dan berhak mendapatkan gelar Sarjana S.Kom.

Dosen Pembimbing 1,	Dosen Pembimbing 2,
Dr. Kestrilia Rega Prilianti, M.Si.	**Hendry Setiawan, ST, M.Kom.,**
NIP. 20120035	**NIP. 20100006**

Dekan Fakultas Sains dan Teknologi

Dr. Eng. Romy Budhi, ST, M.T.
NIP. 20070035

LEMBAR PENGESAHAN TUGAS AKHIR

ANALISIS KINERJA TENSORRT DAN OPENVINO SEBAGAI *INFERENCE FRAMEWORK* UNTUK PEMODELAN *DEEP LEARNING*

Oleh:

LOUIS AXEL TJOKRO ATMODJO

NIM. 312110008

dari:

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI DAN DESAIN

UNIVERSITAS MA CHUNG

Telah dinyatakan lulus dalam melaksanakan Tugas Akhir sebagai syarat kelulusan dan berhak mendapatkan gelar Sarjana S.Kom.

Dosen Pembimbing 1,

Dr. Kestrilia Rega Prilianti, M.Si.
NIP. 20120035

Dosen Pembimbing 2,

Hendry Setiawan, ST, M.Kom.,
NIP. 20100006

Dekan Fakultas Sains dan Teknologi

Dr. Eng. Romy Budhi, ST, M.T.
NIP. 20070035

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan dari Tugas Akhir saya dengan judul "Analisis Kinerja TensorRT dan OpenVINO sebagai *Inference Framework* untuk Pemodelan *Deep Learning*" adalah benar benar hasil karya intelektual mandiri tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Jakarta, 13 Januari 2024

Louis Axel Tjokro Atmodjo

ANALISIS KINERJA TENSORRT DAN OPENVINO SEBAGAI *INFERENCE FRAMEWORK* UNTUK PEMODELAN *DEEP LEARNING*

Louis Axel Tjokro Atmodjo, Kestrilia Rega Prilianti, Hendry Setiawan
Universitas Ma Chung

Abstrak

Deep Learning berusaha meniru kemampuan manusia melalui *Artificial Neural Network* untuk melakukan *task* yang lebih kompleks dari sekedar mengambil keputusan. *Deep learning* yang lebih kompleks, tentunya membutuhkan data dan komputasi lebih kompleks juga. *Inference framework* mampu mengoptimalkan *deep learning* agar lebih efisien tanpa mengorbankan akurasi. Namun, tiap model *deep learning* membutuhkan *inference framework* yang tepat agar mendapatkan performa optimal.

Metode *computer vision* dengan *training dataset* dilakukan terhadap *dataset* citra yang dikumpulkan dalam pencahayaan redup hingga sangat terang menjadi dua buah model *deep learning*, yaitu YOLOv8n dan YOLOv11n yang kemudian dioptimalkan menjadi *inference framework* TensorRT yang memiliki keunggulan dalam pengolahan dengan GPU dari NVIDIA dan OpenVINO yang unggul dengan pengolahan menggunakan CPU dari Intel. Pengujian statistik *T-Test* dan *Two-Way* ANOVA digunakan dalam penelitian ini untuk melihat performa kedua *inference framework* secara terukur dan ilmiah.

TensorRT memiliki rata-rata *inference speed* sebesar 0.006 detik dan rata-rata *model load time* selama 0.0007 detik sehingga sesuai untuk penggunaan *real-time processing* dan unggul daripada OpenVINO. Sedangkan OpenVINO memiliki rata-rata *memory consumption* sebesar 617841.9 MB dan rata-rata *model size* sebesar 6 MB sehingga sesuai untuk penggunaan yang memiliki keterbatasan pada perangkat, karena unggul dari TensorRT. *Accuracy* dari kedua *inference framework* dan tiap model menunjukkan bahwa YOLOv11n lebih unggul sebesar 1% dari YOLOv8n dan OpenVINO mampu mengoptimalkan model yang lebih kompleks dengan lebih baik.

Kata kunci : *deep learning*, *inference framework*, OpenVINO, TensorRT

PERFORMANCE ANALYSIS OF TENSORRT AND OPENVINO AS INFERENCE FRAMEWORK FOR DEEP LEARNING MODELS

Louis Axel Tjokro Atmodjo, Kestrilia Rega Prilianti, Hendry Setiawan
Universitas Ma Chung

Abstract

Deep Learning seeks to mimic human abilities through Artificial Neural Network to perform more complex tasks than just simple task decision-making, such as object recognition. The more complex deep learning, requires larger data and more sophisticated computations. Inference framework can optimize the deep learning models to become more efficient without compromising the accuracy of models. However, each deep learning model requires the correct inference framwork to achieve optimal performance.

Computer vision with training dataset applied to images dataset that captured under low to very bright light conditions, resulting in two deep learning models YOLOv8n and YOLOv11n. Then these models were optimized using TensorRT inference framwork which excels in using GPU with NVIDIA and OpenVINO which superior in working with CPU and Intel. Statistical analysis, such as T-Test and Two-Way ANOVA were utilized in this study to evaluate the performance of both inference framworks in a measurable and scientific manner.

TensorRT has an average inference speed of 0.006 seconds and an average model load time of 0.0007 seconds, making it suitable for real-time processing and superior to OpenVINO in this regard. On the other hand, OpenVINO has an average memory consumption of 617841.9 MB and an average model size of 6 MB, making it more appropriate for devices with limited resources, as it outperforms TensorRT. The accuracy of both inference frameworks and each model demonstrates that YOLOv11n surpasses YOLOv8n by 1%, while OpenVINO also give better results at optimizing more complex models.

Keywords : *deep learning, inference framework*, OpenVINO, TensorRT

Kata Pengantar

Puji syukur dipanjatkan kehadirat Tuhan Yang Maha Esa karena atas rahmat dan restu-Nya sehingga tugas akhir dengan judul "Analisis Kinerja TensorRT dan OpenVINO sebagai *Inference Framework* untuk Pemodelan *Deep Learning*" ini dapat diselesaikan dengan baik. Laporan ini disusun untuk menjelaskan hasil pengerjaan Tugas Akhir yang telah selesai dilaksanakan. Tugas Akhir merupakan salah satu mata kuliah wajib bagi mahasiswa dari Prodi Teknik Informatika Universitas Ma Chung Malang sebagai salah satu prasyarat kelulusan.

Pada kesempatan ini, penulis menyampaikan ucapan terima kasih sebesar-besarnya kepada seluruh pihak yang telah memberikan bantuan dan dukungan kepada penulis dalam proses pengerjaan Tugas Akhir hingga selesai. Ucapan terima kasih disampaikan kepada seluruh pihak terkait yang telah membantu, mendukung, dan membimbing kegiatan Tugas Akhir hingga selesai, di antaranya :

1. Kepala Departemen ITE dari PT. XYZ yang telah menyediakan tempat untuk pelaksanaan penelitian,
2. Bapak Dr. Eng. Romy Budhi, ST, M.T. selaku Dekan Fakultas Teknologi dan Desain Universitas Ma Chung,
3. Bapak Hendry Setiawan, ST, M.Kom, selaku Kepala Program Studi Teknik Informatika dan Dosen Pembimbing 2 Tugas Akhir,
4. Ibu Dr. Kestrilia Rega Prilianti, M.Si. selaku Dosen Pembimbing 1 Tugas Akhir,
5. Seluruh Tim *Data Science* selaku Pembimbing Lapangan,
6. Keluarga terkasih yang telah memberikan dukungan dan semangat selama pengerjaan Tugas Akhir ini,
7. Serta teman-teman yang telah memberikan dukungan untuk menyelesaikan pengerjaan Tugas Akhir.

Laporan ini disusun berdasarkan hasil penelitian di PT. XYZ selama enam bulan dengan judul "Analisis Kinerja TensorRT dan OpenVINO sebagai *Inference Framework* untuk Pemodelan *Deep Learning*".

Jakarta, 13 Januari 2024

Louis Axel Tjokro Atmodjo

Daftar Isi

Daftar Tabel

Daftar Gambar

BAB I
Pendahuluan

1.1 Latar Belakang

Kecerdasan buatan atau yang sering dikenal dengan *Artificial Intelligence* (AI) merupakan teknologi terbaru yang menjadi revolusi teknologi yang sangat pesat dalam beberapa tahun terakhir. Teknologi *artificial intelligence* pertama kali dicetuskan pada 1956 oleh John McCarthy yang membuat memulai riset mengenai sistem atau mesin yang dapat berpikir layaknya manusia. *Artificial intelligence* bermula pada sebuah *rule-based system* yang berjalan pada *task* yang sederhana. Pada beberapa tahun terakhir, *artificial intelligence* sudah mampu melakukan *task* yang lebih kompleks dengan kecepatan yang tinggi, algoritma perhitungan yang kompleks, serta pengolahan data dalam jumlah besar. Kemampuan *artificial intelligence* yang sangat baik ini hingga saat ini mampu diaplikasikan pada *task* yang lebih kompleks juga dan dapat menyesuaikan dengan berbagai bidang seperti kesehatan, keuangan, otomasi, pengolahan bahasa, pembuatan teks, pengenalan objek, hingga prediksi dan pengambilan keputusan.

Artificial intelligence yang dapat diaplikasikan pada beragam bidang ini juga dapat diintegrasikan pada perangkat-perangkat sederhana seperti ponsel genggam dan menjadi *personal assistant*. Dengan ini, *artificial intelligence* menjadi sebuah sistem yang dapat membantu manusia yang mudah diakses dalam melakukan *task* untuk mengoptimalkan kinerja manusia pada hal-hal lain yang membutuhkan perhatian lebih. AI yang berusaha meniru kecerdasan manusia ini membutuhkan data dalam jumlah besar untuk dijadikan pengetahuan dasar atau *knowledge base* bagi sistem AI tersebut.

Pada beberapa tahun terakhir, *Machine Learning* (ML) yang menjadi salah satu cabang ilmu dari *artificial intelligence* menjadi populer karena kemampuannya untuk mengajarkan komputer untuk mengambil keputusan berdasarkan data yang diberikan. Melalui algoritmanya, *machine learning* dapat mempelajari pola-pola yang terdapat pada data-data untuk melakukan prediksi atau rekomendasi. *Machine learning* memiliki beberapa dasar metode untuk mengajarkan mesin untuk memiliki

kecerdasan, seperti *reinforcement learning*, *supervised learning*, *unsupervised learning*. Metode-metode pengajaran mesin tersebut diberlakukan sesuai dengan kebutuhan masing-masing untuk mendapatkan hasil optimal.

Deep Learning (DL) merupakan cabang lanjutan dari *machine learning* yang secara mendalam memaksimalkan potensi *artificial intelligence*, hingga sampai pada meniru struktur dan fungsi dari otak manusia sebagai kecerdasan manusia untuk mendapatkan kecerdasan buatan melalui *Artificial Neural Networks*. Melalui *deep learning*, sistem *artificial intelligence* mampu melakukan *task* yang lebih kompleks daripada *machine learning* yang berfokus pada pengenalan pola dan data. *Deep learning* mampu melakukan *task* seperti pengenalan objek atau citra melalui *Computer Vision*, pengolahan Bahasa melalui *Natural Language Processing*, dan hal lainnya dengan akurasi yang lebih baik daripada model milik *machine learning*. Model milik *deep learning* mampu mempelajari data secara otomatis dengan baik tanpa campur tangan manusia. Namun, kecerdasan dan akurasi ini tentunya datang dengan tantangan besar juga, seperti kebutuhan data yang besar hingga proses komputasi yang sangat berat.

Inference Framework merupakan *framework* yang membantu model *deep learning* yang sangat besar agar dapat digunakan dengan lebih mudah tanpa mengorbankan akurasi pada aplikasinya secara nyata. *Inference framework* memiliki kemampuan untuk meningkatkan kemampuan model *deep learning* untuk penggunaan secara *real-time* dan membuat proses *inference* lebih efektif pada berbagai *hardware platform* sesuai kebutuhan. Hingga saat ini, terdapat beberapa *inference framework* yang populer dengan keahliannya yang terpolarisasi, yaitu TensorRT yang merupakan hasil pengembangan tim NVIDIA dan OpenVINO yang merupakan hasil pengembangan tim Intel. Kedua *inference framework* memiliki *base architecture* yang berbeda secara signifikan, dimana TensorRT dioptimalkan secara khusus untuk melakukan komputasi pada *Graphic Processing Unit* (GPU) milik NVIDIA, sedangkan OpenVINO dirancang khusus untuk melakukan komputasi pada *Central Processing Unit* (CPU). Kedua *inference framework* ini sangatlah penting dalam aplikasinya secara nyata, dimana kecepatan, efektivitas, hingga akurasi menjadi kunci dari keberhasilan sistem *Artificial Intelligence*.

Melalui penelitian yang telah dilakukan Zhou dan Yang (2022), didapatkan hasil bahwa model ONNX yang dikonversi menjadi TensorRT mampu meningkatkan kemampuan *real-time inference* tanpa mengorbankan akurasi. Penelitian lain yang telah dilakukan oleh Zhongyi, *et al* (2019) menunjukkan bahwa OpenVINO mendapatkan hasil memuaskan pada kemampuannya melakukan *detection* secara *real-time* ketika menggunakan model ringan seperti MobileNet, ResNet, dan SqueezeNet. Namun, belum ada penelitian terbaru mengenai kemampuan *inference framework* pada model yang lebih modern saat ini, yaitu YOLO. YOLOv8 mampu melampaui akurasi YOLO terdahulu karena kombinasi pada arsitekturnya (Reis, *et al*, 2024) . Hingga saat ini YOLO telah mengeluarkan YOLOv11 yang dikenalkan mampu memberikan akurasi lebih baik dari YOLOv8. Penelitian kali ini akan berfokus pada analisis kinerja dari TensorRT milik NVIDIA dan OpenVINO milik Intel sebagai sebuah *inference framework* untuk pemodelan dari *Deep Learning*.

1.2 Identifikasi Masalah

Melalui latar belakang masalah yang telah disampaikan, dapat diperoleh identifikasi masalah bahwa *inference framework* milik TensorRT dan OpenVINO memiliki ciri khas masing-masing dengan variasi kinerja yang berbeda, sehingga diperlukan penelitian komprehensif untuk mengetahui kinerja secara *real-time* antara keduanya. Selain itu, penggunaan GPU dan CPU menjadi tantangan baru untuk menemukan performa *inference framework* yang sesuai pada aplikasi secara nyata.

1.3 Batasan Masalah

Berikut merupakan batasan masalah yang ditetapkan pada Tugas Akhir ini:

a) Jenis model yang digunakan dalam penelitian ini adalah YOLOv8 dan YOLOv11
b) Ukuran model yang digunakan pada penelitian ini adalah ukuran *nano* (n)
c) Optimasi model berfokus pada *model conversion*
d) *Hyperparameter tuning* untuk *training* model pada penelitian ini disamakan

1.4 Rumusan Masalah

Bagaimana kinerja TensorRT dan OpenVINO sebagai *inference framework* untuk pemodelan *Deep Learning* pada kasus perhitungan atau *counting box* secara *real-time*.

1.5 Tujuan

Tujuan dari dilakukannya penelitian ini adalah untuk mengetahui perbandingan kinerja TensorRT dan OpenVINO sebagai *inference framework* untuk pemodelan *Deep Learning*.

1.6 Manfaat

Manfaat yang dapat diperoleh dalam penelitian ini adalah sebagai berikut:

1. Bagi praktik industri, manfaat yang didapatkan adalah mengetahui alternatif format model yang menggunakan CPU untuk melakukan *object detection* secara *real-time*.
2. Bagi penulis, maanfaat yang didapatkan adalah memperluas wawasan mengenai ragam format model dalam YOLO untuk melakukan *object detection* secara *real-time*.
3. Bagi Universitas Ma Chung, khususnya Program Studi Teknik Informatika, manfaat yang didapatkan adalah dapat mempersiapkan lulusan yang kompeten dan siap kerja dengan memberikan bekal kepada mahasiswa melalui proses pembelajaran selama Tugas Akhir.

1.7 Luaran

Luaran dalam penelitian ini penjelasan mengenai perbandingan kinerja TensorRT dan OpenVINO sebagai *inference framework* untuk pemodelan *deep learning* beserta beserta analisis serta dilanjutkan dengan publikasi ilmiah ke jurnal.

1.8 Sistematika Penelitian

Sistematika penulisan proposal Tugas Akhir ini dibagi menjadi tiga bab seperti berikut ini.

Bab I Pendahuluan

	Bab ini terdiri dari latar belakang, identifikasi masalah, rumusan masalah, Batasan masalah, tujuan penelitian, manfaat peneltian, luaran tugas akhir, dan sistematika penelitian.
Bab II	Tinjauan Pustaka Bab ini berisi urutan sistematis terkait literatur yang digunakan dalam proses penyusunan laporan Tugas Akhir sehingga diperoleh landasan teori terkait dengan TensorRT dan OpenVINO sebagai *inference framework*, teknik optimasi model, dan pemodelan *deep learning*.
Bab III	Metodologi Penelitian Bab ini menjelaskan mengenai tahapan pengerjaan serta analisis dari penelitian yang akan dilakukan. Tahapan pengerjaan meliputi identifikasi masalah, studi literatur, pembuatan data, *training* model, *export* model, hingga perbandingan kinerja.
Bab IV	Hasil dan Pembahasan Bab ini membahas mengenai hasil dari penelitian yang telah dilakukan untuk menganalisis perbandingan kinerja TensorRT dan OpenVINO sebagai *inference framework* untuk pemodelan *deep learning*.
Bab V	Kesimpulan dan Saran Bab ini berisi simpulan dari hasil penelitian yang telah dilakukan serta saran mengenai yang mungkin dapat memperbaiki kinerja atau penelitian selanjutnya.

BAB II
Tinjauan Pustaka

2.1 *Artificial Intelligence*

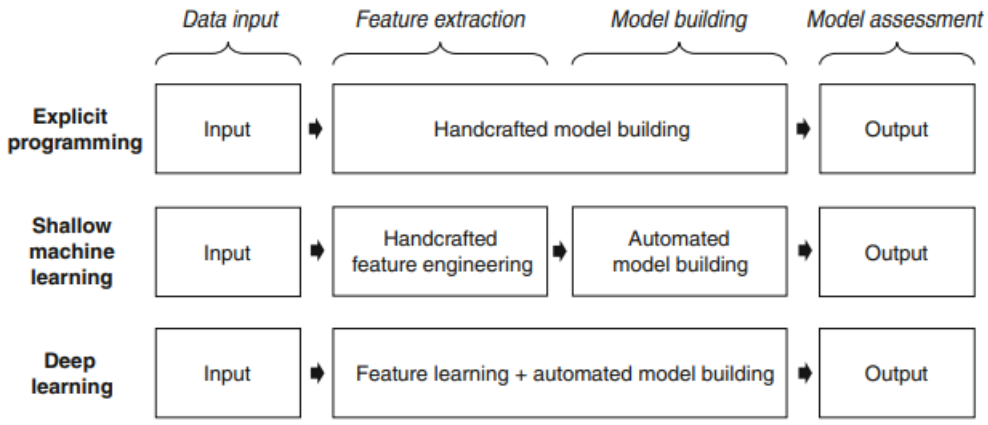
Artificial Intelligence atau dalam bahasa Indonesia berarti kecerdasan buatan merupakan kecerdasan yang ditanamkan pada suatu sistem sehingga dapat diatur sedemikian rupa. Penelitian mengenai *artificial intelligence* telah dimulai pada tahun 1956 oleh John McCarthy dan Marvin Minsky yang menjalankan penelitian selama 8 bulan, yaitu *Dartmouth Summer Research Project on Artificial Intelligence* (DSRPAI) di kampus Dartmouth, New Hampshire (Haenlein, M., Kaplan, A., 2019). Tujuan dari diadakannya DSRPAI adalah untuk mengumpulkan para peneliti untuk akhirnya dapat membangun sebuah mesin yang memiliki kemampuan mensimulasikan kecerdasan manusia. Pada dasarnya *artificial intelligence* ini ditanamkan pada sebuah komputer, agar komputer dapat menjadi cerdas untuk dapat meniru beberapa perilaku manusia seperti perhitungan, pengolahan bahasa, pengetahuan, identifikasi, pemecahan masalah hingga pengambilan suatu keputusan. Dalam jurnalnya, Haenlein, M. Dan Kaplan, A. juga menjelaskan bahwa kecerdasan buatan adalah sebuah sistem yang mampu untuk menafsirkan data eksternal dan belajar dari data tersebut serta menggunakan hasil pembelajarannya untuk mencapai tujuan tertentu (2019).

Latar belakang dari *artificial intelligence* adalah sebuah sistem yang melakukan pengambilan keputusan hingga pemikiran berdasarkan sebuah *reason* atau *knowledge base* dan sebuah *rule-based system*. Namun, sistem sederhana ini sangat terbatas pada beberapa penggunaan yang lebih kompleks dan membutuhkan penyesuaian yang meluas pada skala tertentu. Hingga saat ini, banyak sekali *task* yang dapat dilakukan oleh *artificial intelligence* seperti pengenalan objek pada gambar, melakukan prediksi, dan lain-lain. Kemampuan *artificial intelligence* yang paling populer saat ini adalah *Generative Artificial Intelligence* bersamaan dengan *Large Language Model* yang dapat melakukan interaksi dengan manusia seperti tanya jawab, membuat kesimpulan dari suatu paragraf, hingga hal lain. Kemampuan-kemampuan ini muncul berkat adanya pengembangan dari sistem atau

ide dasar dari *artificial intelligence* sendiri melalui *machine learning* dan *deep learning* yang terus berkembang dengan sangat pesat hingga saat ini. *Artificial intelligence* telah menunjukkan kemampuannya sebagai alat bantu yang berharga dalam melakukan beragam *task* mulai dari analisis, memberikan wawasan melalui motif, dan gaya bahasa yang dapat dilewatkan oleh manusia (Swathi, M., Dhayalakrishnan, R., 2024).

2.1.1 *Machine Learning*

Machine Learning merupakan bagian dari *artificial intelligence* yang mencakup ilmu dalam membuat algoritma yang akan membuat mesin mempelajari data dan terus berkembang secara pertahap. Kemampuan *machine learning* ini berbeda dengan program atau sistem pada umumnya yang perlu memerlukan campur tangan manusia. Konsep dasar dari sebuah *machine learning* adalah mesin atau sistem yang mampu mempelajari data hingga mengenali pola yang terdapat pada data untuk akhirnya membuat sebuah prediksi atau pengambilan keputusan untuk data baru diluar data sebelumnya. Terdapat beberapa metode dalam *machine learning*, seperti *supervised learning*, *unsupervised learning*, *semi-supervised learning*, dan *reinforcement learning*. *Machine learning* dengan beragam metodenya masih kesulitan dalam menangani data yang kompleks untuk melakukan *feature extraction* atau mencari pola dan hubungan pada data. Sehingga, *machine learning* masih membutuhkan bantuan manusia dalam melakukan *feature extraction* yang akan digunakan nantinya untuk membuat algoritmanya.



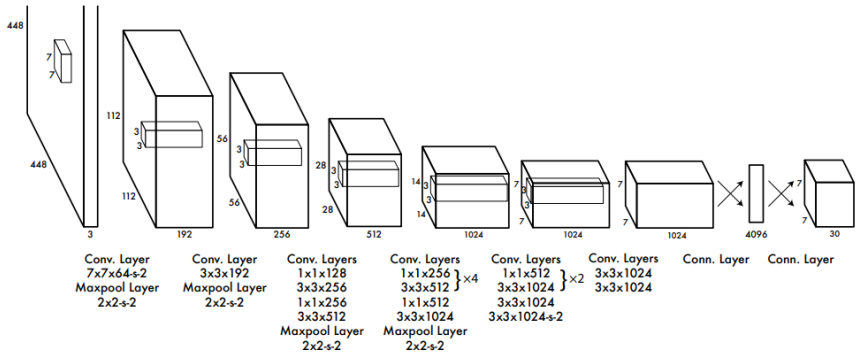
Gambar 2.1 Ilustrasi Pembuatan Model (Janiesch, C, *et al.* 2021)

2.1.2 *Deep Learning*

Perkembangan lebih lanjut dari *machine learning* adalah *Deep Learning* yang berusaha menirukan jaringan saraf manusia. Berbeda dengan *machine learning* yang lebih sederhana, *deep learning* memiliki lapisan dalam jumlah banyak untuk menjawab kekurangan dari *machine learning*. *Deep learning* memiliki lapisan atau *layer* ini adalah bagian yang memuat informasi pola hingga relasi yang terdapat pada data. Ciri khas *deep learning* adalah kemampuannya untuk mengenali pola hingga relasi pada data secara mandiri dan memperbaiki hasil keluarannya. Pada algoritma sederhana, jika ingin melakukan perubahan pada model, dibutuhkan banyak penyesuaian, sedangkan *deep learning* hanya perlu melakukan penyesuaian pada parameter (Hao, 2018).

2.1.2.1 YOLO

You Only Look Once (YOLO) merupakan sebuah algoritma *Deep learning* yang umum digunakan sebagai pendeteksi objek yang memiliki kelebihan dalam hal kecepatan dan akurasi. Algoritma ini dapat mendeteksi objek secara *real-time* dengan pendekatan *Convolutional Neural Network* (CNN). CNN mampu membagi suatu citra menjadi beberapa bagian (dalam *pixel*) dan memprediksi setiap kotak yang kemudian dibandingkan dengan probabilitas yang diprediksi. Model dari YOLO menggunakan *deep learning* untuk meningkatkan kecepatan dan akurasi dalam melakukan deteksi dan klasifikasi objek.

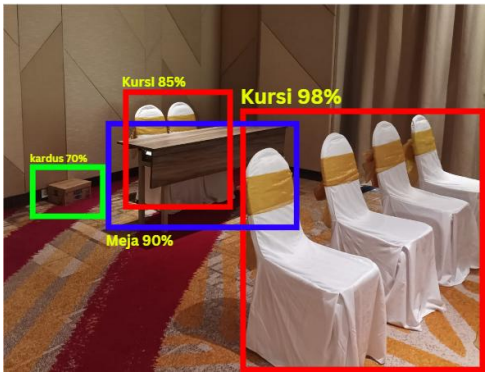


Gambar 2.2 Arsitektur YOLO (Redmon, J., *et al*. 2016)

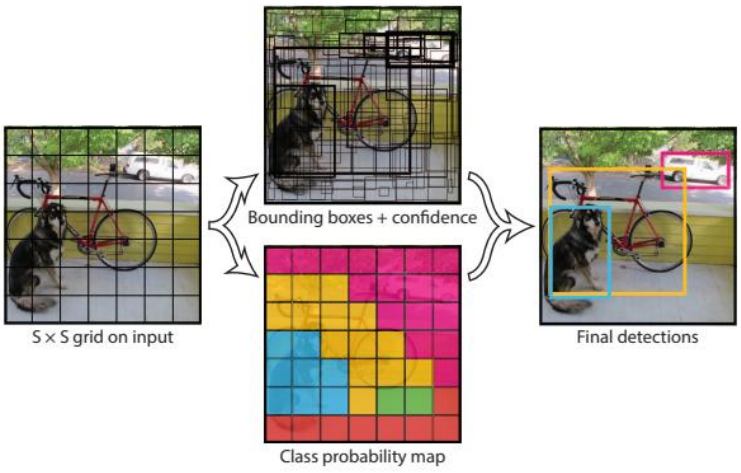
Arsitektur dari YOLO menggunakan 24 *convolutional layers* dengan 2 *fully connected layers*. Secara bergantian 1 x 1 *convolutional layers* mengurangi fitur

yang terdapat pada *layer* sebelumnya. YOLO juga melakukan *pre-train* pada *convolutional layers* pada ImageNet *classification* sebesar setengah dari resolusi citra dan kemudian menggandakannya untuk deteksi (Redmon, J., *et al*. 2016)

YOLO dapat memberikan informasi koordinat dari *bounding box* atau kotak pembatas (area yang dideteksi sebagai objek) dan mengklasifikasikan objek yang dideteksi. *Bounding box* memberikan informasi x dan y sebagai pusat dari *bounding box* dan *height* dan *width* sebagai ukurannya, serta nilai dari probabilitasnya. Algoritma yang bekerja dalam model YOLO adalah dengan membagi citra yang diberikan menjadi S x S dengan S adalah jumlah dari sel.



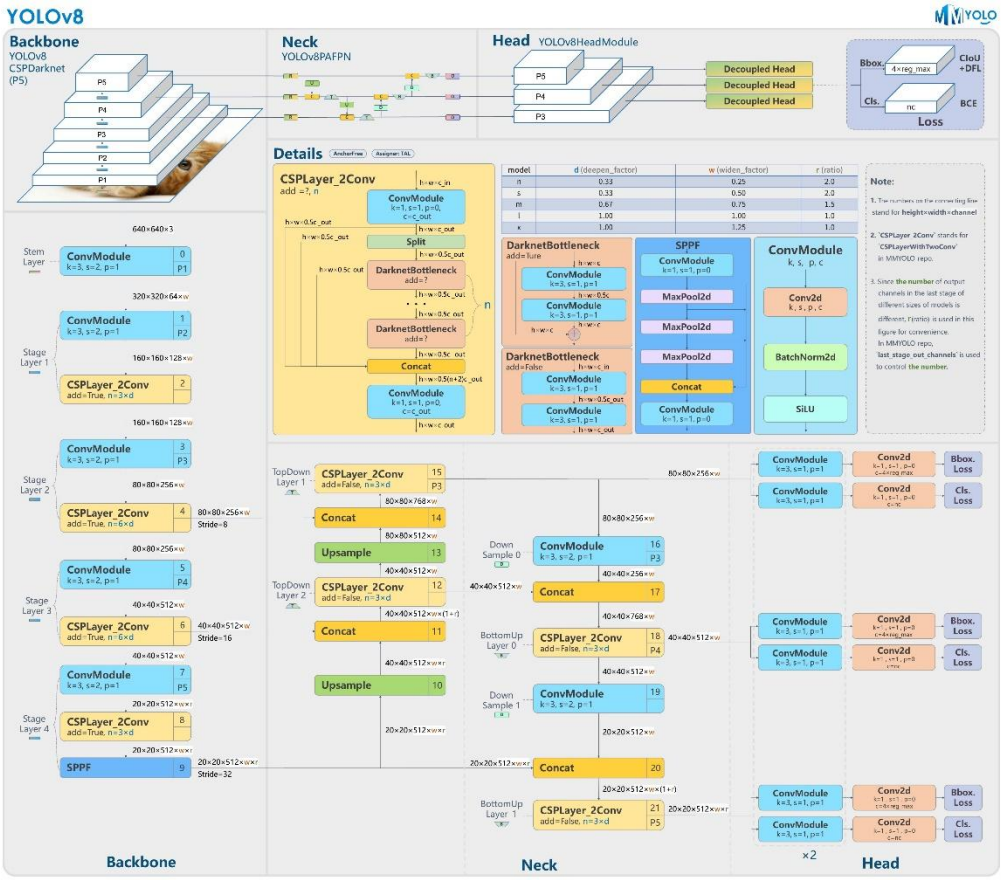
Gambar 2.3 Ilustrasi *Bounding Box*



Gambar 2.4 Pembagian Citra S x S (Redmon, *et al*. 2016)

YOLOv8 merupakan salah satu perkembangan dari YOLO yang dirilis pada bulan Januari tahun 2023 dan dikembangkan oleh tim dari perusahaan Ultralytics,

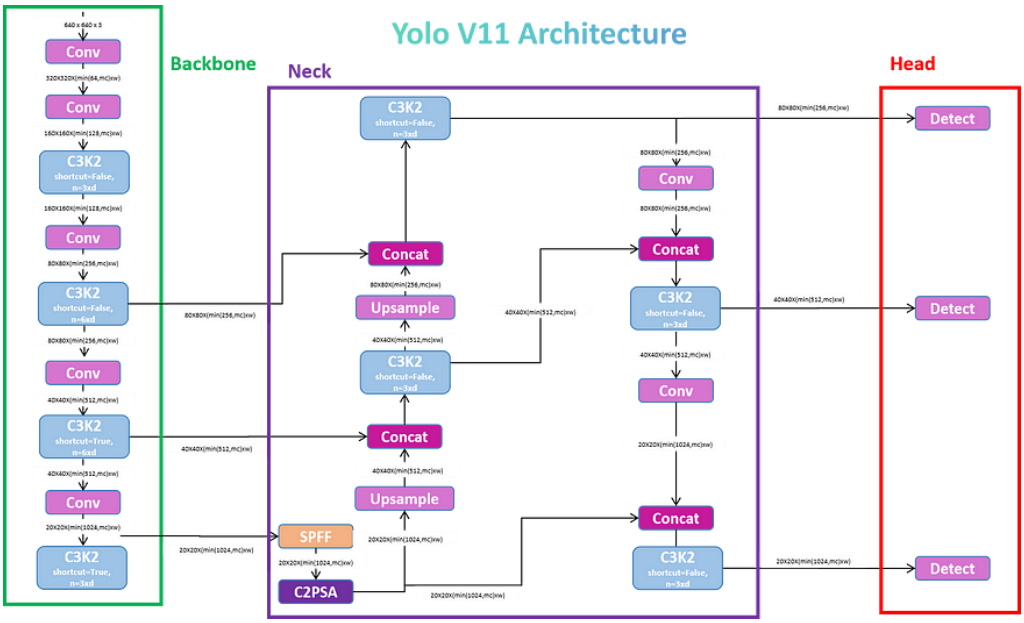
sebuah perusahaan teknologi yang juga telah mengembangkan YOLOv5 sebelumnya. Terdapat 5 jenis versi YOLOv8 yang tersedia, yaitu YOLOv8n (*nano*), YOLOv8s (*small*), YOLOv8m (*medium*), YOLOv8l (*large*), dan YOLOv8x (*extra-large*). Keseluruhan versi YOLOv8 menunjukkan ukuran dan kemampuan model YOLOv8, mulai dari YOLOv8n yang terkecil serta ringan, hingga YOLOv8x yang terbesar dan berat. Namun, secara keseluruhan YOLOv8 mampu mendukung kemampuan *computer vision* seperti *object detection*, *image segmentation*, *pose estimation*, *tracking*, dan *classification* (Terven, J., *et al*. 2023).



Gambar 2.5 Arsitektur YOLOv8 (Huang, H., 2024)

YOLOv11 merupakan versi terbaru yang juga dikembangkan oleh tim Ultralytics dan dirilis pada September 2024. Terdapat 5 versi YOLOv11 yang tersedia, yaitu YOLOv11n (*nano*), YOLOv11s (*small*), YOLOv11m (*medium*), YOLOv11l (*large*), dan YOLOv11x (*extra-large*). YOLOv11 memiliki kemampuan untuk meningkatkan akurasi melalui mAP sebesar 22% dengan jumlah

parameter lebih sedikit ketika dibandingkan dengan YOLOv8m. Perkembangan terbaru arsitektur YOLOv11 pada *backbone*, *neck* dan peningkatan pada kemampuan *feature extraction* dapat membawa memberikan akurasi yang lebih baik yang berpengaruh pada diagnosis kanker (Awad, *et al*, 2024).



Gambar 2.6 Arsitektur YOLOv11 (Rao, S. N., 2024)

2.2 Implementasi Artificial Intelligence

Industri 4.0 berarti bahwa industri telah berevolusi dan saat ini beralih menuju yang keempat kalinya. Revolusi industri yang pertama terjadi pada 1784 di Inggris, ketika mesin uap pertama kali ditemukan untuk menggantikan tenaga manusia. Revolusi dari industri yang keempat ini diperkasai oleh negara Jerman yang menjadikan industri sebagai kebijakan pembangunannya. Pada akhirnya, beberapa negara lain juga menyusul menerapkan industri 4.0 ini dengan nama-nama lain seperti *Smart Industry*, *Internet of Things*, dan lain-lain. Sesuai dengan revolusi industri pertama di Inggris, revolusi industri keempat ini juga masih melihat bidang industri sebagai porosnya dengan contohnya adalah kecepatan produksi, fleksibilitas produksi, dan hal lain yang berkaitan dengan bidang industri manufaktur dan pada akhirnya dapat membawa perkembangan dari sebuah negara. Dilihat dari industri penerapannya, maka bidang industri manufakturlah yang

memiliki peran besar (Prasetyo dan Sutopo, 2018). Disisi lain, terdapat bidang industri otomotif, kesehatan, energi, logistik, hingga industri pertanian yang terkena dampak revolusi industri 4.0 dan menuntut hal seperti kecepatan dan fleksibilitas.

Seiring perkembangan teknologi, muncul a*rtificial intelligence* atau kecerdasan buatan yang cerdas dan dapat membantu berbagai keperluan yang ada. Dilihat dari perspektif positf, tantangan industri yang kompleks justru membuka peluang bagi kemampuan unik yang dimiliki teknologi kecerdasan buatan dibandingkan dengan alat dan pendekatan konvensional (Arinez, *et al*, 2020). Kemampuan dari kecerdasan buatan dalam memaksimalkan industri sangat beragam, dari meningkatkan produktivitas, fleksibilitas, keamanan, kualitas hingga mengurangi biaya produksi.

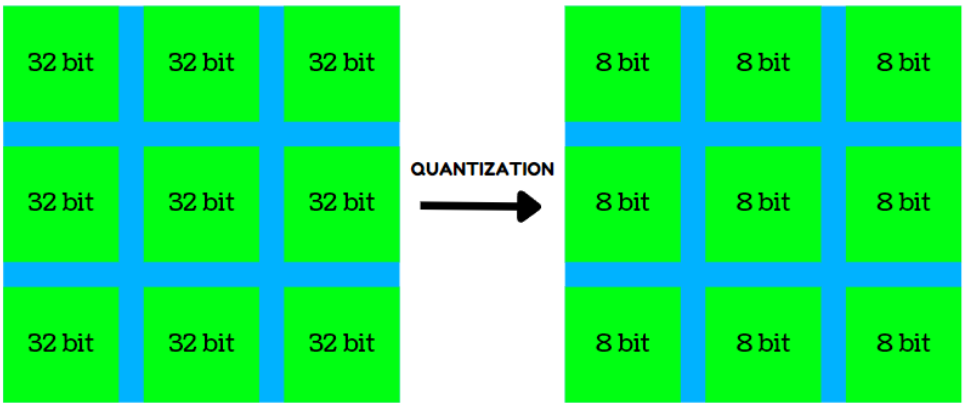
Ilmu kecerdasan buatan telah digunakan dalam lini produksi, kesehatan hingga logistik dan pertanian, contohnya adalah optimisasi produksi, deteksi cacat pada produk, identifikasi hingga prediksi *downtime* dari suatu mesin, diagnosis penyakit dari citra *x-ray*, *smart watch* dengan sistem pemantauan dan rekomendasinya, rekomendasi rute logistik, rekomendasi *warehouse stock management*, dan deteksi dini penyakit pada tumbuhan melalui citra satelit serta sistem irigasi cerdas untuk industri pertanian. Hasil dari implementasi kecerdasan buatan ini telah terbukti meningkatkan produktivitas dan kualitas dari industri. Keuntungan lain dari kecerdasan buatan dalam industri adalah membuat industri berjalan dengan lebih mudah, cepat dan fleksibel untuk perputaran industrinya. Menurut Lubis (2021), implementasi kecerdasan buatan pada pengerjaan manusia dapat menghasilkan kinerja optimal dengan waktu proses yang cepat dan hasil maksimal.

2.2.1 Model *Optimization*

Sebuah model pada *deep learning* tentunya memiliki layer yang sangat kaya akan pengetahuan atau *knowledge base* untuk mencapai kecerdasan yang sesuai dengan keinginan atau *training* pada *dataset* yang telah dilakukan sebelumnya. Model dari YOLO yang memiliki kemampuan *object detection* secara *real-time* dengan akurasi tinggi tentunya akan memiliki *knowledge base* yang juga sangat luas, sehingga ukuran dari model ini cukup besar. Proses *training* bertujuan agar

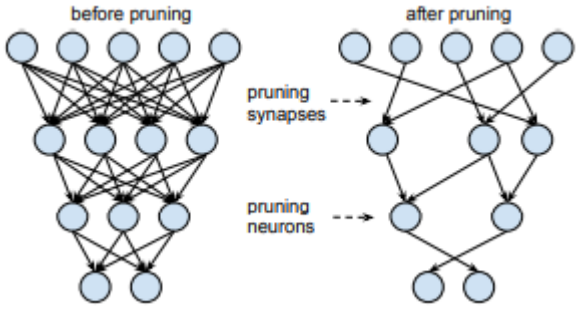
model memiliki *knowledge base* sesuai dengan yang diharapkan, sedangkan untuk menggunakan model tersebut agar dapat melakukan *task* seperti *prediction* pada data baru dibutuhkan satu buah *tools* yang dapat membantu *deployment* dari model tersebut. *Tools* yang dapat membantu proses *deployment* model agar dapat melakukan *task* yang diinginkan dari model disebut dengan *inference frameworks*.

Sebuah *inference frameworks* memiliki kemampuan untuk melakukan model *deployment* dan model *optimization* yang dibutuhkan untuk meningkatkan efektivitas model. Dalam penggunaan industri, tentunya penggunaan model YOLO tidak semata-mata melakukan *task* atau *inference* satu kali saja, tetapi secara berulang dengan menggunakan *edge device* yang memiliki spesifikasi atau sumber daya terbatas. Dari sinilah muncul kebutuhan untuk melakukan optimasi untuk mengurangi ukuran dari model agar lebih ringan tanpa mengurangi *knowledge* yang dimiliki oleh model tersebut, sehingga model dapat berjalan dengan efisien dan maksimal. Terdapat beberapa teknik yang dapat diterapkan untuk mengoptimalkan sebuah model tanpa mengganggu kinerja model tersebut. *Quantization* merupakan teknik untuk merubah angka yang memiliki keakuratan tinggi atau *high-precision floating-point* yang dalam *neural networks* menjadi angka yang memiliki keakuratan lebih rendah atau *low-precision* seperti contohnya 0.56 menjadi 78. Dengan mengurangi tingkat keakuratan sebuah angka, maka model tentunya menjadi lebih sederhana dan berukuran lebih kecil yang dapat mengurangi penggunaan *memory consumption* selama proses *inference*.



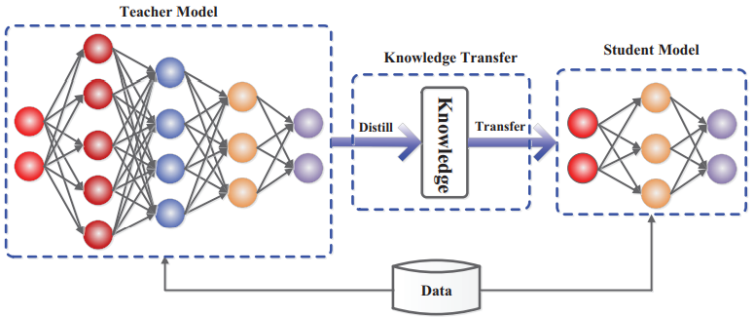
Gambar 2.7 Ilustrasi *Quantization*

Pruning merupakan teknik untuk menghapus jaringan neuron atau filter yang dianggap tidak penting pada model. Tujuan dari *pruning* adalah untuk mengurangi kompleksitas model dan mempercepat proses *inference*. *Pruning* menghilangkan jaringan neuron yang memiliki beban kecil atau kurang mempengaruhi *output* model agar mendapatkan ukuran model yang lebih ringan dengan *memory consumption* yang lebih sedikit. Keuntungan yang didapat dari teknik ini selain ukuran yang lebih kecil adalah dengan struktur model yang lebih sederhana dapat mencegah terjadinya ketidakmampuan model mendeteksi data baru (*overfitting*) karena memiliki *knowledge* yang lebih *general*. Teknik *pruning* dapat mempercepat proses *inference* tanpa mengorbankan akurasi dari model.



Gambar 2.8 Ilustrasi *Pruning*

Knowledge distillation adalah teknik untuk *transfer knowledge* dari model besar kepada model yang lebih kecil. Solusi optimisasi yang disampaikan Hinton, *et al* disebut sebagai *distillation* yang berarti menggunakan *output* dari model besar beserta rincian jaringannya untuk melatih model yang lebih kecil agar dapat meniru hasil dari model yang besar (2015).



Gambar 2.9 Ilustrasi *Knowledge Distillation*

Fusing layers merupakan teknik yang mampu menggabungkan beberapa *layer* menjadi satu untuk mengurangi *memory consumption* agar performa *inference* dapat ditingkatkan. Ditemukan bahwa menggabungkan beberapa *layer* yang paling sama antara satu dan lainnya dapat memberikan performa yang setara dengan model aslinya (O'Neill, *et al*. 2020).

Teknik *quantization, pruning, knowledge distillation, fusing layer* saat ini tidak perlu dilakukan secara manual dan terpisah dikarenakan sudah terdapat beberapa *tools* untuk *inference* yang telah disediakan. Beberapa *frameworks* popular yang sudah tersedia untuk melakukan *inference* adalah TensorRT oleh NVIDIA, ONNX-runtime oleh Microsoft, OpenVINO oleh Intel, MXNet oleh Apache. Seluruh *inference frameworks* dikembangkan sesuai dengan algoritma masing-masing yang unik. *Inference framework* yang cukup popular adalah milik NVIDIA, yaitu TensorRT yang melakukan optimisasi menggunakan *hardware* GPU (*Graphic Processing Unit*) milik NVIDIA yang dikenal dengan kecepatan inferensi, sehingga sesuai jika digunakan dalam kebutuhan komputasi dengan performa tinggi. Pada sisi lainnya, OpenVINO yang dikembangkan oleh Intel memiliki kemampuan komputasi pada CPU (*Central Processing Unit*). Kedua *inference framework* memiliki kelebihan masing-masing untuk kebutuhan yang berbeda. Perbedaan arsitektur *inference framework* dan penggunaan *hardware* yang cukup signifikan pada proses komputasi milik TensorRT dan OpenVINO menjadikan keduanya perbandingan yang sesuai untuk mengetahui kemampuan *inference framework* berdarkan *hardware* yang berbeda dan sesuai untuk *real-time detection*.

Masing-masing *inference framework* memiliki arsitektur dan algoritma tertentu untuk mencapai tujuan tertentu. Sehingga pada setiap *inference framework* dapat menghasilkan hasil yang berbeda untuk satu kasus yang sama. TensorRT yang merupakan *inference framework* dengan dasar pengembangan GPU NVIDIA tentunya akan lebih optimal ketika digunakan untuk kasus spesifik yang membutuhkan GPU dalam komputasinya dan lebih baik jika menggunakan GPU milik NVIDIA. Di sisi lain, penggunaan OpenVINO akan lebih optimal pada kasus komputasi dengan CPU dan CPU milik Intel. Kelebihan dari *inference framework*

milik TensorRT adalah peningkatan performa sebesar 3.5 jika dibandingkan dengan model yang tidak dioptimasi, *Frame per Second* (FPS) yang lebih tinggi dalam melakukan *real-time detection*, optimasi *memory* untuk penggunaan *Embedded System*, mendukung format *Open Neural Network Exchange* (ONNX) untuk kemampuan penggunaan pada berbagai platform, dan kemampuan untuk melakukan penyesuaian melalui *Compute Unified Device Architecture* (CUDA) (Al Ghadani, *et al*, 2020). Penggunaan TensorRT sebagai *inference framework* dengan segala kelebihannya, tentunya juga memiliki kekurangan. Menurut Assunção, *et al* (2022), kekurangan dari TensorRT adalah kehilangan akurasi dari 75% menjadi 64% pada mIOU (*mean Intersection over Union*) ketika menggunakan *depth multiplier* sebesar 0.5, kompleksitas *hyperparameter tuning*, ketergantungan pada perangkat NVIDIA yang memiliki CUDA, kebutuhan *memory* spesifik yang disebabkan oleh konfigurasi yang dilakukan, dan *overfitting* karena kemampuan model dalam melakukan optimisasi yang berlebihan. *Inference framework* OpenVINO milik Intel juga menawarkan kelebihan seperti, performa optimal ketika menggunakan perangkat Intel, peningkatan performa sebesar 3,3 kali pada perangkat Intel dengan *quantization* sebesar 75%, *multi-platform compatibility* mulai dari *edge device* hingga *cloud* dan berbagai *framework* lain seperti TensorFlow dan PyTorch, *high throughput* dengan *low latency* yang membuatnya sesuai dengan *real-time detection*, hemat daya atau dapat dijalankan pada perangkat dengan daya yang rendah, hingga kemudahaan penggunaanya (Ahn, *et al*, 2023). OpenVINO juga memiliki kekurangan, seperti yang dijelaskan oleh Mani, *et al* (2024) seperti, penggunaannya akan terlimit jika tidak menggunakan perangkat Intel, kurangnya fleksibilitas pada penggunaan *framework* selain TensorFlow dan PyTorch, rumit pada konfigurasi, tidak semua model *support* OpenVINO.

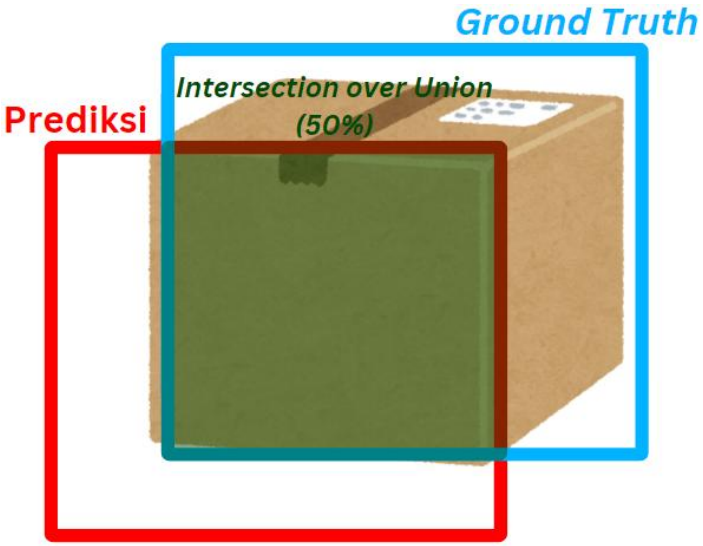
Terdapat beberapa indikator perbandingan untuk mengetahui performa masing-masing *inference framework* adalah dengan melihat *inference speed*, *memory consumption*, *power consumption*, *model size*, *loading time*, dan *accuracy*. Namun, tidak semua indikator sesuai dan memiliki kualitas yang sama ketika melakukan perbandingan performa *inference framework* pada tiap kasus. Pada penelitian ini, terdapat lima indikator yang dirasa sesuai dan tepat dengan

kebutuhan *real-time object detection*. Kelimanya adalah *inference speed, memory consumption, model size, loading time*, dan *accuracy* yang melihat dari nilai mAP atau *mean Average Precision*.

1. *Inference speed* merupakan satu indikator penting untuk menilai efektivitas *inference framework* dalam melakukan *task* secara *real-time*. *Inference speed* merupakan waktu yang dibutuhkan *inference framework* dalam memproses dan menghasilkan prediksi dari data masukkan yang diberikan yang pada umumnya diukur dengan satuan *millisecond*. Dalam konteks penelitian ini, *inference speed* dievaluasi berdasarkan kemampuan *inference framework* untuk menyelesaikan *task object detection* pada satu data masukkan. Semakin cepat atau semakin kecil *inference speed* dalam menyelesaikan *task* pada data masukkan, semakin efektif *inference framework* dalam mendukung *real-time object detection*.
2. *Memory consumption* menjadi indikator perbandingan yang penting dalam menilai skalabilitas dan efektivitas sebuah *inference framwork* secara khusus jika digunakan pada *hardware* dengan spesifikasi yang lebih terbatas sesuai dengan kebutuhan. *Memory consumption* berarti jumlah sumber daya memori yang digunakan oleh *inference framework* selama proses *inference* dilakukan yang berdampak secara langsung pada kemampuan system melakukan *task*. Pada penelitian ini, *memory consumption* dapat dievaluasi dengan melihat penggunaan RAM (*Random Access Memory*) mulai dari *inference framework* menerima data masukkan hingga menyelesaikan *task object detection*. Semakin rendah atau kecil *memory consumption* selama proses *inference* dilakukan, maka semakin efektif *inference framework* tersebut dalam mendukung *real-time object detection*.
3. *Model size* dapat menjadi indikator perbandingan berikutnya yang sesuai dalam menilai skalabilitas *inference framework* yang secara khusus digunakan pada *hardware* dengan spesifikasi terbatas untuk mengoptimalkan infrastruktur penyimpanannya. *Model size* merupakan

ukuran *file* dari *inference framework* yang dapat diukur dengan satuan *megabyte* (MB) atau *gigabyte* (GB) yang dapat menjadi indikator penting dari kompleksitas dan efektifitas suatu *inference framework*. Dalam penelitian ini *model size* dapat dievaluasi berdasarkan ukuran penyimpanannya. Semakin rendah atau kecil *model size* yang dimiliki *inference framework*, maka semakin baik efektifitas *inference framework* dalam melakukan *inference* tanpa mengorbankan kebutuhan *hardware*.

4. *Loading time* sebuah *inference framework* berperan penting dalam menunjukkan efektifitas *inference framework* secara khusus pada penggunaan dalam *task real-time object detection*. *Loading time* merupakan indikator yang menunjukkan waktu yang diperlukan untuk memuat *inference framework* ke dalam memori sebelum melakukan *inference* pada data masukkan. *Loading time* dapat dipengaruhi oleh arsitektur, metode *optimization* yang dilakukan oleh *inference framework*. Dalam penelitian ini *loading time* dapat dievaluasi melalui kecepatan *inference framework* dimuat pada memori sebelum *inference* mulai dilakukan. Semakin rendah atau cepat *loading time* suatu *inference framework*, maka semakin efektif suatu *inference framework* untuk melakukan *task real-time object detection* terutama pada kebutuhan kecepatan inisiasi.
5. *Accuracy* tentunya menjadi indikator utama dalam melihat kemampuan *inference framework* melakukan *task* yang diberikan pada berbagai kondisi yang ada. *Accuracy* yang dimiliki oleh *inference framework* biasanya diukur melalui *mean Average Precision* (mAP) yang menunjukkan kemampuan *inference framework* dalam melakukan *task* yang diberikan. Pada penelitian ini, mAP digunakan sebagai indikator utama dalam mengukur performa *inference framework* pada presisi *Intersection over Union* (IoU) 50%. IoU 50% dapat diartikan bahwa hasil prediksi *bounding box* mengalami tumpang tindih dengan bounding box dari ground truth deteksinya sebesar minimal 50%.



Gambar 2.10 Ilustrasi *Intersection over Union* (IoU) 50%

Semakin besar atau tinggi nilai dari mAP(50%) suatu *inference framework*, maka semakin akurat *inference framework* dalam melakukan *task* yang diberikan secara konsisten dengan *threshold* IoU 50%.

Penelitian ini dilakukan dengan melihat performa *inference framework* pada dua *hardware* yang berbeda dengan tujuan untuk memperkaya wawasan mengenai performa masing-masing *inference framework* sesuai dengan desain arsitekturnya. *Hardware* yang pertama adalah komputer dengan CPU Intel(R) Gold Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz yang memiliki *storage* sebesar 2.5 *terrabyte* berupa *Hard Disk Drive* (HDD) CPU 64 *core* dan *Random Access Memory* (RAM) sebesar 130 *gigabyte*. *Hardware* kedua adalah milik NVIDIA dengan spesifikasi CPU Arm® Cortex®-A78AE v8.2 64-bit yang memiliki 12 *core* dengan *storage* sebesar 256 *gigabyte* dan memiliki RAM sebesar 64 *gigabyte* 256-bit LPDDR5 yang memiliki kecepatan 204.8 *gigabyte*/*second* dengan GPU NVIDIA *Ampere architecture* dengan 2045 NVIDIA® CUDA® *cores* dan 64 *Tensor cores*.

Kedua *hardware* memiliki perbedaan yang signifikan dengan ciri khas masing-masing. *Hardware* pertama milik Intel yang prosesnya difokuskan pada CPU miliknya, sedangkan *hardware* kedua milik NVIDIA yang *computing background* miliknya difokuskan pada GPU. Python 3.8 digunakan pada kedua

hardware untuk mengetahui kinerja masing-masing *inference framework* untuk mendapatkan perbandingan yang setara dengan *operating system* LINUX UBUNTU.

2.3 Pengujian Statistik

Dalam melakukan perbandingan yang membutuhkan pengambilan kesimpulan berdasarkan data numerik yang tersedia, diperlukan suatu metode pengujian yang terukur untuk mengetahui hasil perbandingan secara terukur dengan metode sesuai. Pengujian statistik merupakan suatu metode pengujian yang terukur secara ilmiah untuk menarik kesimpulan berdasarkan data yang diberikan. Pengujian statistik memungkinkan peneliti untuk melihat dan menarik kesimpulan mengenai perbedaan antar data hingga mengetahui hubungan antar variabel secara pada data atau antar data. Metode statistik ini dapat memberikan hasil yang signifikan berdasarkan ketentuan yang ditentukan secara mandiri. Tujuan dari pengujian statistik adalah untuk melakukan komparasi dan melihat interaksi antara kelompok data yang mencakup perbedaan rata-rata, interaksi hingga variabel dan dampak perlakuan tertentu pada kelompok data tertentu. Selain itu, pengujian statistik juga penting dilakukan untuk mengetahui reliabilitas data agar dapat dipertanggungjawabkan, mengetahui jenis data, memberikan pemahaman atas faktor tertentu terhadap data tertentu yang berujung pada kebutuhan pengambilan keputusan atau kesimpulan berdasarkan analisis data yang objektif. Metode uji statistik memiliki beberapa teknik pengujian yang dapat digunakan untuk mengambil kesimpulan (Ramadhani, dkk, 2022).

Terdapat dua metode pengujian statistik yang umum digunakan, yaitu *T-Test* dan ANOVA (*Analysis of Variance*). Uji statistik uji T dapat digunakan untuk mencari tahu apakah terdapat perbedaan signifikan pada data dengan cara membandingkan nilai rata-rata dari satu atau dua data yang tersedia. Dalam uji T sendiri terdapat beberapa jenis bergantung dengan karakteristik dari data. Uji T yang pertama adalah *one-sample T-Test*, untuk jenis uji T ini tujuan utamanya adalah mengetahui apakah terdapat perbedaan signifikan antara satu nilai dengan rata-rata dari satu set data, contohnya adalah untuk mengetahui apakah terdapat perbedaan signifikan dari rata-rata tinggi satu kelas dengan tinggi seorang siswa.

Jenis uji T berikutnya adalah *independent T-Test* yang biasa digunakan untuk mengetahui apakah terdapat perbedaan signifikan antara dua set data dengan syarat kedua data tidak memiliki hubungan. Contoh dari *independent T-Test* ini adalah untuk mengetahui apakah terdapat perbedaan signifikan antara rata-rata kecepatan berlari laki-laki dengan rata-rata kecepatan berlari wanita, dimana kedua data tidak memiliki relasi. Berikutnya, terdapat *paired / dependent T-Test* yang bertujuan untuk mengetahui apakah terdapat perbedaan signifikan dari rata-rata dua set data yang memiliki relasi. *Paired / dependent T-Test* dapat digunakan untuk mengetahui apakah terdapat perbedaan rata-rata berat badan harian seseorang ketika sebelum berolahraga secara rutin dan sesudah berolahraga secara rutin, dimana terdapat relasi antara keduanya yaitu pada subjek pengamatan. Pada uji statistik *independent T-Test*, diperlukan data dan jenis *variance* dari kedua data independen. Data *variance* merupakan data yang menunjukkan homogenitas antara setiap nilai pada data dengan rata-rata maupun nilai tengah pada data. Apabila *variance* menunjukkan terdapat homogenitas pada data, maka dapat diartikan bahwa varians data pada kelompok tidak berbeda secara signifikan. Uji homogenitas pada data atau uji *variance* dapat dijadikan sebagai penentu berikutnya untuk metode *independent T-Test*. Apabila data disebut homogen melalui uji *variance*, maka dapat langsung menggunakan *independent T-Test*, sebaliknya jika data tidak homogen, maka akan menggunakan *Welch's T-Test*. Untuk melakukan uji homogenitas data, dapat dilakukan uji *levene*.

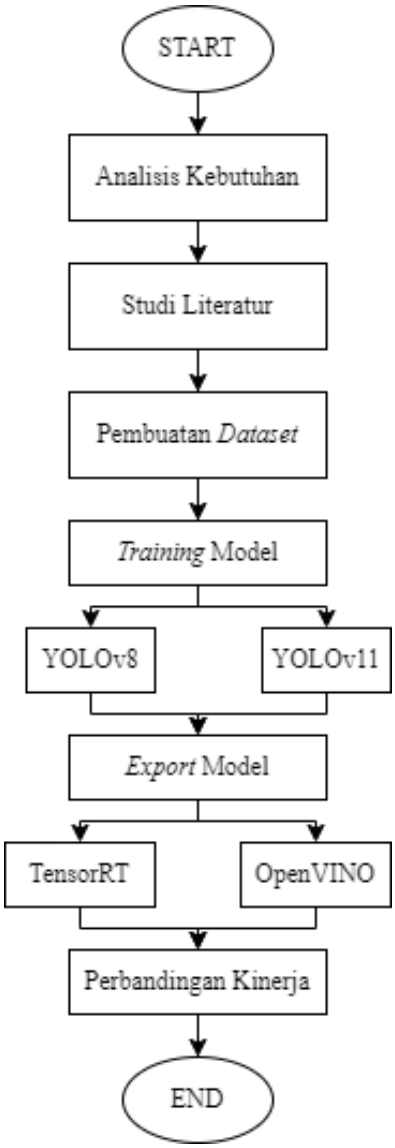
ANOVA yang merupakan singkatan dari *Analysis of Variance* yang merupakan salah satu uji statistik dengan yang bertujuan untuk menguji perbedaan secara signifikan pada rata-rata antara dua faktor atau lebih pada data. Pada dasarnya, uji ANOVA memiliki tujuan yang sama dengan uji T, tetapi uji ANOVA memiliki keunggulan dalam kemampuannya untuk menguji perbedaan secara signifikan pada kelompok data yang memiliki faktor uji. Pada uji ANOVA, terdapat dua jenis uji yang dapat dilakukan, yaitu *One-Way* ANOVA dan *Two-Way* ANOVA. Uji *One-Way* ANOVA merupakan pengujian statistik yang serupa dengan *T-Test* dengan dua set data, tetapi lebih kompleks pada metode statistiknya, sehingga hasil yang disajikan uji ANOVA juga lebih kompleks dan disajikan dalam

bentuk tabel atau yang dikenal dengan tabel F. Perhitungan statistik yang di lakukan pada *One-Way* ANOVA dengan menghitung rata-rata atau *mean*, standar deviasi, standar *error* rata-rata, nilai maksimum, nilai minimum, selang kepercayaan rata-rata, dan uji *levene* untuk setiap set data. Pada metode *Two-Way* ANOVA dapat dilakukan pada kebutuhan perhitungan yang lebih kompleks dan dilakukan untuk mengetahui perbedaan secara signifikan antar set data yang diberikan pada faktor tertentu. Sederhananya, *Two-Way* ANOVA tidak hanya menguji perbedaan signifikan pada variabel dalam set data, tetapi juga melakukan pengujian antara set data untuk mengetahui interaksi antar set data yang diberikan untuk mengetahui faktor yang berinteraksi satu sama lain pada data. Pada penelitian ini digunakan dua buah *inference framework*, yaitu TensorRT dan OpenVINO sebagai faktor pertama dan pada dua model yang berbeda, yaitu YOLOv8n dan YOLOv11n sebagai faktor kedua, sehingga perlu dikaji juga secara mendalam pada data untuk tiap kategori atau indikator perbandingan serta antar model pada tiap *inference framework* untuk melihat interaksi pada model untuk tiap *inference framework*. Berdasarkan kondisi datadan kebutuhan kajian, maka dalam penelitian ini akan digunakan *Two-Way* ANOVA yang mampu menguji faktor tertentu dalam set data dan variabel antar set data. Sama dengan pengujian statistik menggunakan *T-Test* perlu dilakukan pengujian terhadap homogenitas data dengan menggunakan uji *levene*.

BAB III
Analisis dan Perancangan Sistem

3.1 Alur Penelitian

Proyek Tugas Akhir ini memiliki tujuan untuk mengetahui perbandingan performa dari *inference framework* yang disediakan oleh OpenVINO yang dikembangkan Intel dan *inference framework* TensorRT yang disediakan oleh NVIDIA. Alur penelitian dari proyek ini dapat dilihat pada gambar 3.1.



Gambar 3.1 Alur Penelitian

3.2 Analisis Kebutuhan

Dalam penelitian untuk menganalisis perbandingan performa antara *inference framework* milik OpenVINO dan TensorRT, dibutuhkan analisis agar penelitian dapat berjalan dengan lancar dan optimal. Analisis kebutuhan yang dilakukan dalam menganalisis perbandingan performa kedua *inference framework* mencakup analisis kebutuhan dari sisi pengguna dan peneliti.

3.2.1 Kebutuhan Pengguna

Analisis kebutuhan dapat dimulai dengan mengidentifikasi masalah utama yang dihadapi, yaitu keperluan mencari alternatif yang efisien dari TensorRT milik NVIDIA. TensorRT merupakan *framework* dikenal optimal dalam melakukan *inference* menggunakan GPU, terutama untuk keperluan seperti *object detection*, hingga *counting box* secara *real-time*. Namun, penggunaan GPU yang intensif menjadi tantangan, terutama karena konsumsi sumber daya yang besar, yang terkadang tidak sebanding dengan kebutuhan atau keterbatasan infrastruktur pada khususnya dalam dunia industri. Oleh karena itu, penting untuk menemukan solusi lain yang lebih *cost saving* dalam penggunaan sumber daya maupun secara komputasi, dengan tetap mampu mempertahankan performa yang dibutuhkan.

Salah satu alternatif yang dapat dipertimbangkan adalah framework *inference* yang tidak hanya bergantung pada GPU, tetapi juga memungkinkan penggunaan CPU secara optimal. Intel menawarkan solusi komputasi berbasis CPU dengan nama OpenVINO yang dapat menjadi alternatif menarik untuk menggantikan TensorRT dalam konteks *object detection*, hingga *counting box* secara *real-time*. Penggunaan CPU ini diharapkan mampu mengurangi ketergantungan pada GPU dan memberikan efektivitas yang lebih baik dalam hal *cost saving*.

Framework yang berbasis CPU ini juga memiliki potensi untuk diterapkan di berbagai lingkungan yang mungkin tidak memiliki akses ke GPU dengan spesifikasi tinggi. Dengan begitu, solusi ini dapat memperluas jangkauan penggunaan teknologi *counting box* dalam skenario *real-time* yang lebih luas. Di sisi lain, penerapan framework CPU tentu juga memiliki tantangan, terutama dalam hal memastikan bahwa performa tetap optimal tanpa kehilangan kecepatan atau akurasi yang diharapkan dan mampu dipenuhi oleh GPU. Oleh karena itu,

penelitian ini bertujuan untuk mengevaluasi dan membandingkan kinerja *inference framework* alternatif milik OpenVINO dengan TensorRT. Hasil dari penelitian ini diharapkan dapat memberikan gambaran yang jelas mengenai apakah solusi berbasis CPU dapat menjadi pilihan yang layak untuk keperluan *counting box* secara *real-time*. Implementasi dari penelitian ini juga akan diguanakan oleh PT. XYZ sebagai bahan pertimbangan mengenai penggunaan *inference framework* utnuk optimisasi kinerja *counting box* secara *real-time* untuk membantu meningkatkan sistem perhitungan yang saat ini diterapkan.

3.2.2 Kebutuhan Eksperimen

Dalam melakukan penelitian, penulis membutuhkan dukungan perangkat keras dan perangkat lunak agar penelitian dapat berjalan dengan baik dan optimal. Pemilihan perangkat keras atau *hardware* perlu dilakukan dengan benar sesuai dengan kondisi yang ada dan kebutuhan dari masing-masing *inference framework*. Kesalahan pemilihan *hardware* dapat mengakibatkan kesalahan analisis dan membuat hasil tidak optimal. Pemilihan perangkat lunak atau *software* juga tidak dapat dilakukan tanpa adanya dasar yang baik. Akibat dari kesalahan pemilihan *software* dapat menyebabkan kegagalan program. Berikut adalah *hardware* dan *software* yang dibutuhkan berdasarkan pemilihan untuk menunjang penelitian ini.

1. Perangkat Keras (*Hardware*) :
    a. Komputer A:
        i. *Processor* : Intel(R) Gold Intel(R) Xeon(R) Gold 5218 2.30GHz
        ii. GPU : 130 *gigabyte*
        iii. *Memory* : *Hard Disk Drive* 2.5 *terrabyte* 64 *core*
        iv. *Operating system* : Linux UBUNTU
    b. Komputer B:
        i. *Processor* : Arm® Cortex®-A78AE v8.2 64-bit 12 *core*
        ii. GPU : 64 gigabyte 256-bit LPDDR5 204.8 *gigabyte*/*second*
        iii. *Memory* : *Solid State Drive* 256 *gigabyte*

iv. *Operating system* : Linux UBUNTU

2. Perangkat Lunak (*software*):
    a. Python3
    b. Visual Studio Code
    c. Supervision
    d. You Only Look Once (YOLO)
    e. Pandas
    f. Numpy
    g. Torch

3.3 Studi Literatur

Pada tahapan studi literatur, akan dilakukan studi mengenai penelitian-penelitian terdahulu, buku hingga jurnal artikel mengenai topik terkait yang sekiranya dapat dijadikan sebagai bahan referensi dalam menyelesaikan permasalahan. Topik-topik pada penelitian terdahulu yang dapat dijadikan sebagai bahan referensi adalah mengenai arsitektur dari model YOLO mulai dari arsitektur dari YOLO hingga perbedaan antara YOLOv8 dengan YOLOv11, teknik-teknik yang dapat dilakukan untuk melakukan optimisasi model, dan arsitektur dari OpenVINO dan TensorRT dalam melakukan optimisasi. Studi literatur terhadap teknik optimasi model dapat dijadikan sebagai acuan dalam mengetahui cara teknik yang dapat dilakukan dalam mengoptimalkan kinerja model. Kajian topik arsitektur YOLO hingga arsitektur dari YOLOv8 dan YOLOv11 difokuskan untuk mengetahui perbedaan arsitektur dari kedua model. Studi literatur yang berkaitan dengan arsitektur dari OpenVINO dan TensorRT akan berfokus menyelesaikan permasalahan yang terdapat pada penggunaan CPU dan GPU dalam meningkatkan kinerja model. Studi literatur terhadap perbandingan juga dibutuhkan agar penelitian ini dapat terukur dengan baik. Proses studi literatur ini ditujukan untuk mempelajari langkah serta metode yang sesuai dengan penelitian agar dapat dilakukan analisis yang selaras sehingga mendapatkan hasil yang baik. Studi literatur akan mencakup penjelasan mengenai kedua *inference framework* dalam mengoptimalkan kinerja model. Ringkasan dari studi literatur yang dilakukan terdapat pada tabel 3.1.

Tabel 3.1 Studi Literatur

No	Topik	Pengetahuan	Temuan
1.	YOLO	YOLO merupakan *object detector* yang akurat.	YOLO mampu mempertahankan kemampuannya pada kasus *real-time*. (Redmon, *et al*, 2016).
2.	YOLOv8	YOLOv8 memiliki arsitektur yang serupa dengan pendahulunya.	Arsitektur milik YOLOv8 yang menggabungkan FAN dan PAN serta peningkatan *post-processing* membuat YOLOv8 lebih unggul dari pendahulunya (Reis, *et al*, 2024)
3.	YOLOv11	YOLOv11 merupakan versi terbaru dari YOLO yang menawarkan akurasi tinggi dengan parameter lebih sedikit.	YOLOv11 dapat memberikan akurasi yang sedikit lebih baik daripada YOLOv8 pada diagnosis kanker (Awad, *et al*, 2024).
4.	*Model Optimization*	*Model optimization* dibutuhkan untuk mengaplikasikan *deep learning* pada *real-world application*.	Teknik optimasi seperti *quantization*, *pruning*, dan *knowledge distillation* mampu meningkatkan efektivitas model tanpa mengorbankan performanya (Kim, *et al*, 2021).

Tabel 3.1 Lanjutan Studi Literatur

No	Topik	Pengetahuan	Temuan
5.	OpenVINO	*Inference framework* untuk membantu mengoptimalkan model pada CPU yang dikembangkan oleh Intel	*Model optimization* OpenVINO mampu meningkatkan akurasi dari model (Biswas, *et al*, 2020).
6.	TensorRT	*Inference framework* milik NVIDIA yang mengoptimalkan model pada GPU NVIDIA	TensorRT meningkatkan kemampuan komputasi dan kecepatan model sebesar tiga kali lipat (Chaturvedi, *et al*, 2022)
7.	Indikator perbandingan *inference framework*	Indikator dalam melakukan perbandingan untuk *inference framework*	Beberapa indikator perbandingan dapat digunakan untuk menemukan kekuatan dan kelemahan dari tiap *inference framework* (Pochelu, 2022)

3.4 Pembuatan *Dataset*

Data didapatkan dari PT. XYZ yang merupakan sebuah perusahaan manufaktur. Total data yang digunakan adalah sejumlah 1976 gambar kardus yang terdiri dari 9 jenis *box* yang berbeda. Kesembilan jenis *box* yang dijadikan *dataset* terbagi menjadi 6 buah *dataset* foto kardus milik PT. XYZ yang memiliki identitas material dan tidak memiliki identitas material, serta 3 buah *dataset* foto kardus milik anak perusahaan PT. XYZ yang memiliki identitas material dan multi identitas material.

Proses pengumpulan data dilakukan oleh tim produksi dan tim *office*. Pengambilan data dilakukan dengan mengambil gambar kardus dari berbagai sudut dan tingkat kecerahan yang juga beragam. Anotasi data dilakukan dengan bantuan aplikasi *Computer Vision Annotation Tool* (CVAT) yang memiliki *service* untuk

anotasi data dengan cepat dengan menggunakan 2D *bounding box, polygon, point polylinesi* yang programnya terdapat pada komputer A.

Tahapan anotasi data menggunakan CVAT mebutuhkan beberapa langkah terlebih dahulu hingga akhirnya data dapat digunakan sebagai *dataset* untuk *training* model. Pertama adalah membuat dengan proyek baru dan kemudian menyiapkan kelas-kelas yang akan digunakan dengan *tools* yang sudah disiapkan CVAT. Setelah seluruh data dianotasi, barulah *dataset* dapat digunakan sebagai *dataset training* untuk *modeling*. Dalam penelitian ini, terdapat lima kelas *object* yang dibutuhkan. Kelas-kelas tersebut adalah *box* universal SKU, *box* satu SKU, *universal* SKU, SKU, dan *flag*. Setelah membuat kelas-kelas yang dibutuhkan dalam *object detection* untuk penelitian ini, dilakukan proses *labeling* atau anotasi data dengan bantuan CVAT sesuai dengan kelasnya masing-masing pada setiap *dataset*. Pada penelitian ini, anotasi data kali ini menggunakan *bounding box* dengan bentuk *rectangle* karena sesuai dengan bentuk dari data gambar, yaitu *box*. Tahapan anotasi data yang dilakukan pada keseluruhan gambar dibantu oleh tim *outsource*. Terdapat dua jenis *box* utama dalam penelitian ini, yang pertama adalah *box* dengan satu identitas atau satu SKU yang tercetak pada *box*, dan *box universal* dengan beberapa identitas atau *universal* SKU yang berati terdapat beberapa SKU yang tercetak pada *box*, dimana identitas sebenarnya ditandai dengan *object flag*.

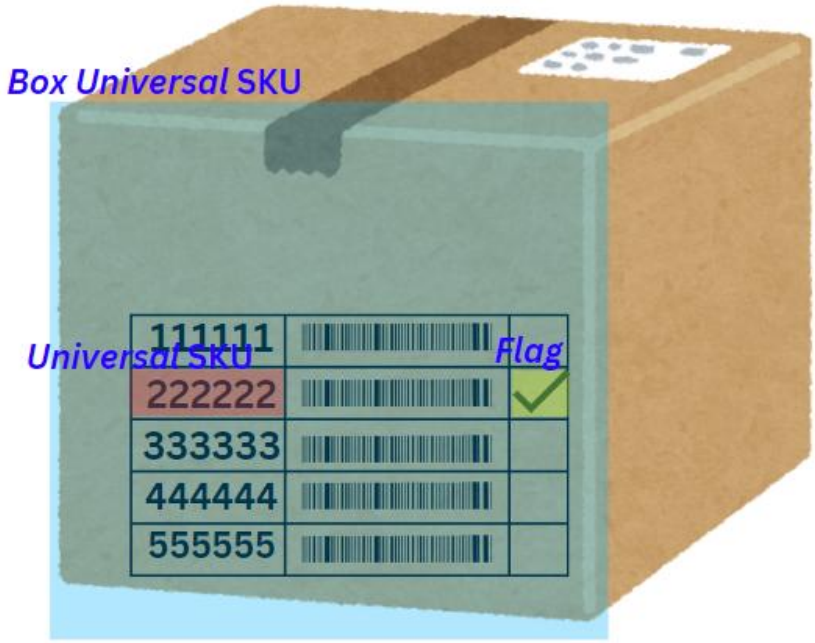
Gambar 3.2 Ilustrasi *Dataset Box* Satu SKU

Gambar 3.3 merupakan ilustrasi dari gambar *box* dengan satu identitas atau satu SKU yang dimiliki oleh PT. XYZ, sedangkan gambar 3.4 merupakan ilustrasi dari gambar *box universal* yang memiliki beberapa identitas atau *universal* SKU. Desain kotak multi SKU dirancang agar lebih efisien saat memproduksi variasi lain dari suatu produk. Untuk *box* dengan SKU universal, terdapat tanda atau *flag* sebagai penunjuk identitas kotak sebenarnya. *Flag* ini menandai kotak yang berisi produk yang sedang diproduksi. Lokasi flag berada di sisi kanan kode batang yang terletak di bagian tengah kotak dengan SKU sebenarnya terletak di sisi kiri dari kode batang tersebut. Desain *box* ini membantu untuk mempermudah identifikasi produk dalam *box* yang sedang diproduksi. *Box* universal ini memungkinkan PT. XYZ memproduksi berbagai produk tanpa perlu mengubah desain kotak secara signifikan. Penggunaan *flag* menjadi kunci dalam membedakan antara satu produk dengan produk lainnya, atau biasanya satu produk yang memiliki varian yang beragam. Efisiensi dalam penggunaan kotak multi SKU meningkatkan produktivitas dan fleksibilitas dalam produksi. Hal ini membuat proses produksi menjadi lebih optimal dan *cost saving*.

Gambar 3.3 Ilustrasi *Dataset Box Universal* SKU

Dataset untuk *train* dan *validation* terbagi menjadi 1580 gambar untuk *train dataset* dan 396 gambar untuk *validation dataset*. *Train* dan *validation* dari seluruh *dataset* menggunakan bantuan YOLOv8 dan YOLOv11 dengan outputnya adalah

best.pt yang merupakan model dari proyek ini. Pada penelitian ini, model yang berupa *best.pt* akan diberlakukan *task export* menjadi *best.openvino* yang merupakan *inference framework* dari OpenVINO untuk *inference* milik Intel dan menjadi *best.engine* yang untuk *inference* menggunakan TensorRT yang merupakan *inference framework* milik NVIDIA. *Train* dan *validation* dilakukan di komputer A karena *dataset* sudah disiapkan pada komputer A. Untuk *task export* model dilakukan pada masing-masing komputer sesuai dengan arsitektur *inference framework*. Gambar 3.5 dan gambar 3.6 merupakan ilustrasi dari hasil anotasi atau gambar yang telah diberi label pada *dataset*.



Gambar 3.4 Ilustrasi Anotasi pada Dataset Box Satu SKU

3.5 *Training* Model

Para tahapan pelatihan model atau *training* model yang akan dilakukan menggunakan komputer A, perlu dilakukan beberapa hal terlebih dahulu. Setelah selesai melakukan anotasi data, perlu dilakukan *split* antara data *training* dan data untuk *test*. Pada penlitian ini digunakan perbandingan 80% untuk data *training* dan 20% untuk data *testing*, sehingga total data untuk *training* adalah 1580 dan total data untuk *testing* adalah 396 data. Tahapan pertama setelah anotasi dan *split* data

adalah melakukan anotasi data yang sudah dijelaskan sebelumnya. Berikutnya adalah melakukan *review* terhadap *dataset* yang sudah dianotasi. *Review dataset* ini berguna untuk memastikan agar model tidak *overfit* dan *dataset* yang digunakan memiliki kualitas yang baik secara ukuran hingga variasi data. Apabila terdapat kekurangan pada *dataset*, maka tahapan pengumpulan data akan dilakukan kembali untuk melengkapi data-data gambar yang kurang bervariasi. Setelah *dataset* disiapkan, perlu dilakukan konversi dari seluruh *dataset* menjadi satu buah *file* yang memiliki format *.yaml* untuk mempermudah melakukan *training* model. Konversi format ini dapat dilakukan dengan menggunakan *tools* bawaan dari CVAT setelah proses anotasi data selesai dilakukan.

Setelah disiapkan *dataset* dengan format *.yaml*, barulah dapat diberlakukan tahapan *training* model pada menggunakan YOLOv8 dan YOLOv11. Proses *training* dilakukan menggunakan komputer A dengan bantuan program python dan *library* seperti *ultralytics* dan YOLO dengan parameter seperti pada tabel 3.2.

Tabel 3.2 *Hyperparameter Tuning*

Parameter	**Penjelasan Singkat**	**Nilai**
imgsize	Mengubah ukuran citra	640
Batch	Untuk pengaturan *batch size*	64
Epochs	Total *epoch*	8000
Pretrained	*Pretrained* model dari COCO	*True*
lr0	*Learning Rate* awal	0.01
lr1	*Learning Rate* akhir	0.1
momentum	Faktor momentum	0.937
weight_decay	L2 untuk menghindari overfit	0.0005
warmup_epochs	*Learning rate* untuk fase *warmup*	3
warmup_momentum	Momentum untuk *warmup*	0.8
warmup_bias_lr	*Learning rate* untuk *warmup*	0.1
box	Beban dari *box loss*	0.05
cls	Beban dari *classification loss*	0.3
iou	*Intersection over Union threshold*	0.2
hsv_h	Penyesuaian *hue*	0.015
hsv_s	Penyesuaian *saturation*	0.7

Tabel 3.2 Lanjutan *Hyperparameter Tuning*

Parameter	Penjelasan Singkat	Nilai
hsv_v	Penyesuaian *value*	0.4
degrees	Rotasi citra	45.0
translate	Transalasi citra	0.2
scale	Skala citra	0.5
shear	Memotong citra	5.0
perspective	Transformasi citra acak	0.0001
flipud	Memutar balikkan citra atas bawah	0.25
fliplr	Memutar balikkan citra kiri kanan	0.1
mosaic	Mengombinasikan 4 citra *train*	0.0
mixup	Mencampur citra	0.0
dropout	*Dropout*	0.2
patience	Jumlah *epoch* tanpa perkembangan	300

Melalui tabel 3.2 dapat dilihat bahwa terdapat proses *preprocessing* yang diberlakukan pada *dataset*. Diantaranya adalah *resize*, penyesuaian warna HSV, *translate, scale, shear, flipud, fliplr*. Tujuan dari proses *preprocessing* pada tahapan *training* model ini adalah untuk memperkaya pengetahuan *knowledge base* dengan menambah variasi pada *dataset*. Parameter *pretrained* diatur menjadi *True* dengan *dataset* dari COCO yang memiliki kelas yang banyak seperti orang, kendaraan mobil, sepeda, dan objek lainnya.

3.6 *Export* Model

Tahapan *export* model bertujuan agar model yang sudah diberlakukan *training* dengan *dataset* sebelumnya dapat digunakan untuk melakukan *task* sesuai dengan keinginan. Dengan memberlakukan *export*, model dapat digunakan untuk *detection, segmentation, classify, pose estimation*, dan *Oriented Bounding Boxes* (OBB) *object detection*. Selain agar dapat digunakan untuk *inference, export* model dapat membuat model menjadi lebih optimal dengan memberlakukan *model optimization* pada model *best.pt* yang didapatkan dari *training* model. Terdapat beberapa teknik optimisasi model yang ada, tetapi sudah terdapat beberapa *tools* untuk mempermudah melakukan *model optimization* yang telah disediakan.

Beberapa *inference framework* yang dapat digunakan sebagai *tools* untuk mempermudah *model optimization* adalah OpenVINO, TensorRT, ONNX, hingga TF Lite. Masing-masing *inference framework* tersebut sudah memiliki metode-metodenya sendiri untuk melakukan *model optimization* dengan cara yang khas dan memiliki tujuan yang berbeda sesuai dengan arsitektur tiap *inference framework.*

Library YOLO yang digunakan dalam penelitian ini telah menawarkan cara mudah untuk melakukan *export* model. YOLO menyediakan *mode* untuk melakukan *export* model menjadi *inference framework* sesuai dengan keinginan dan kebutuhan yang berbeda. Sebagai contohnya untuk *inference framework* pada *mobile device* dapat mengguankan format TF Lite. Tiap versi dari YOLO menyediakan *inference framework* yang berbeda, tetapi terdapat beberapa *inference framework* yang selalu ada, yaitu OpenVINO dan TensorRT. *Inference framework* milik OpenVINO menawarkan arsitektur yang membuat model dapat menggunakan CPU dengan tiga kali lebih cepat, sedangkan TensorRT menawarkan arsitektur yang mampu mendorong kemampuan GPU hingga lima kali lipat. Pada penelitian yang berfokus melihat potensi CPU sebagai alternatif dari GPU, tentunya akan menggunakan *inference framework* dari OpenVINO untuk CPU dan TensorRT untuk mengetahui performa *inference framework* menggunakan GPU.

3.7 Perbandingan Kinerja

Pada tahapan perbandingan kinerja akan dilakukan eksperimen dan membandingkan serta menganalisis hasil dari dua *inference framework* yang berbeda, yaitu OpenVINO dan TensorRT. Kedua *inference framework* akan diimplementasikan menggunakan model yang telah diberlakukan *training* dengan *dataset* yang sama pada YOLOv8 dan YOLOv11. Masing-masing *inference framework* memiliki kemampuan untuk mengubah format model agar dapat diberlakukan *task* lain seperti *prediction, classification,* hingga *detection* dan dapat mengoptimalkan kinerja dari model tersebut. Perbedaan antara OpenVINO dan TensorRT sangat mendasar, terletak pada arsitekturnya. OpenVINO yang dikembangkan oleh Intel mengoptimalkan kinerja model dan dirancang untuk dijalankan pada perangkat milik CPU milik Intel dan FPGA. *Inference framework* TensorRT yang dikembangkan oleh NVIDIA dirancang dan disesuaikan sehingga mampu mengoptimalkan kinerja model pada *hardware* GPU milik NVIDIA.

Perbandingan kinerja pada penelitian kali ini mencakup beberapa hal agar penelitian tetap sesuai dengan keinginan, yaitu untuk mengetahui performa *inference framework* pada kebutuhan *object detection* secara *real-time* khususnya untuk objek *box* yang berjalan pada *belt conveyor*. Indikator perbandingan untuk melihat kinerja dari masing-masing *inference framework* adalah *inference speed*, *memory consumption*, *model size*, *loading time*, dan *accuracy* yang melihat dari nilai mAP atau *mean Average Precision*. Dengan melakukan perbandingan antara *inference framework* dari OpenVINO dan TensorRT menggunakan kelima indikator tersebut, diharapkan dapat memberikan wawasan yang sesuai pada *trade-off* antara performa keduanya berdasarkan *resource* dan arstiketur masing-masing ketika kedua *inference framework* diterapkan pada model YOLOv8 dan YOLOv11 yang telah melalui proses *training* dengan *dataset* yang sama.

Tabel 3.3 Indikator Perbandingan

Indikator	**TensorRT**		**OpenVINO**	
	YOLOv8	**YOLOv11**	**YOLOv8**	**YOLOv11**
Inference speed (s)				
Memory consumption (MB)				
Model size (MB)				
Loading time (s)				
Accuracy (%)				

Indikator *inference speed* akan didapatkan dari melakukan model *validation* menggunakan fungsi *val()* dari *ultralytics* yang dapat memberikan informasi lengkap mulai dari *preprocess, inference speed, loss,* dan *postprocess* tiap citra pada *dataset test*. Indikator *memory consumption* didapatkan dari pengukuran

jumlah *memory* yang digunakan saat program dijalankan dengan bantuan *library psutils* yang memiliki fungsi *Process()* dan memiliki *getter* berupa *memory_info()* dan *Resident Set Size* (RSS) yang memberikan informasi penggunaan *memory* dalam satuan *kilobyte* (KB). Indikator *model size* didapatkan dari melihat ukuran model setelah dioptimisasi dengan cara klik kanan dan memilih *properties* pada model. Indikator *loading time* didapatkan dari memberikan *timestamp* pada *start* dan *end* untuk proses *load model* dengan bantuan *library time* yang memberikan informasi waktu dalam satuan detik. Indikator *accuracy* dengan mAP50 didapatkan dari fungsi *metrics* dan *box* secara spesifik *mAP50* untuk mendapatkan akurasi berdasarkan mAP50.

```python
from ultralytics import YOLO
import time
import psutil

# Load a model
load_time = time.time() # start load time
model = YOLO("path/to/model")  # load an model
end_load = time.time() - load_time # end load time

# Validate the model
metrics = model.val(data='data_validation.yaml')  # no arguments needed, dataset and settings remembered
map50 = metrics.box.map50  # map50
mem_usage = psutil.Process().memory_info().rss #

# show output
print('load time:', end_load)
print('memory:', mem_usage)
print('map50:', map50)
```

Gambar 3.5 *Snapshot* Kode Python Pengukuran Indikator

Gambar 3.5 merupakan potongan kode program python yang digunakan dalam penelitian ini untuk mendapatkan indikator *inference speed*, *memory consumption*, *load time*, dan *accuracy*. Gambar 3.6 merupakan contoh keluaran dari program python yang merupakan hasil dari pengukuran indikator pada penelitian ini.

```
Speed: 1.2ms preprocess, 17.3ms inference, 0.0ms loss, 0.8ms postprocess per image
Results saved to runs/detect/val
load time: 0.002273082733154297
memory: 2428084224
map50: 0.9523533077162062
```

Gambar 3.6 Contoh Hasil Pengukuran Indikator

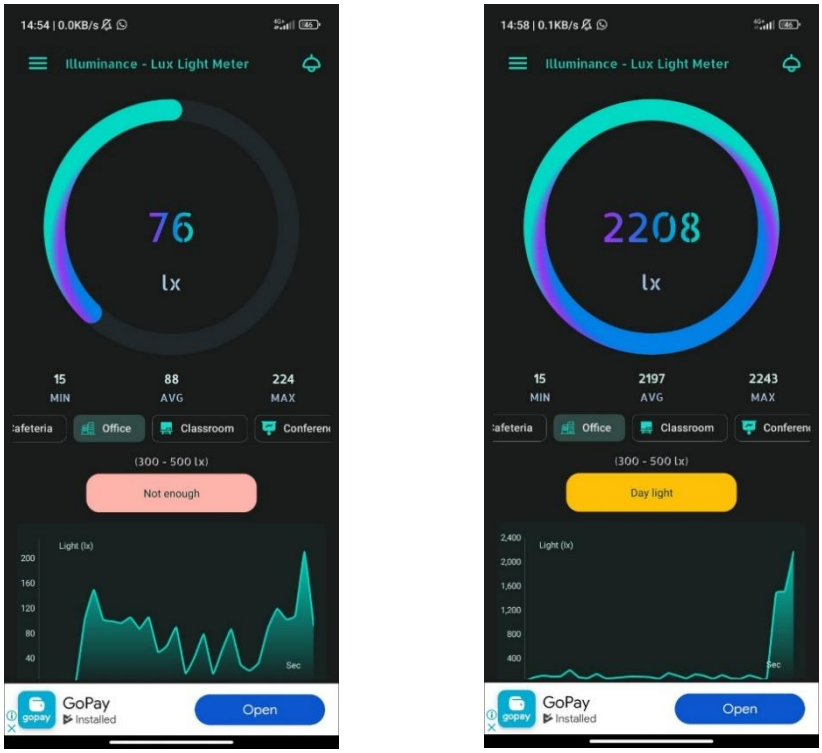
BAB IV
Hasil dan Pembahasan

4.1 Persiapan Perangkat

Perangkat keras yang digunakan merupakan aset milik PT. XYZ yang digunakan sesuai dengan izin yang diperoleh. Komputer A dan komputer B pada saat yang bersamaan digunakan untuk kebutuhan Perusahaan PT. XYZ, sehingga untuk penelitian ini, dibutuhkan koordinasi jadwal. Setelah mendapatkan izin dan jadwal penggunaan komputer A, peneliti baru dapat melanjutkan penelitiannya, mulai dari anotasi *dataset* hingga proses optimisasi model dan eksperimen. Perangkat lunak yang digunakan didapatkan dengan mudah, karena perangkat lunaknya bersifat *open source* yang dapat diakses dengan mudah. Persiapan perangkat lunak dilakukan dengan melakukan instalasi *library* pada kedua perangkat keras.

4.2 *Dataset*

Pembuatan *dataset* dibantu oleh tim produksi di lapangan untuk mengumpulkan data berdasarkan arahan peneliti. Ketentuan *dataset* yang dibutuhkan untuk penelitian ini adalah gambar *box* dengan berbagai sudut dengan pencahayaan ruangan yang cukup variatif dalam keadaan cenderung gelap maupun terang dengan berbagai desain *box* yang tersedia. Variasi pencahayaan dalam melakukan pengumpulan *dataset* ada pada pencahayaan gelap sebesar 76 lux pada pencahayaan normal ruangan di gudang dan area produksi hingga pencahayaan terang sebesar 2208 lux pada pencahayaan normal ruangan kerja menggunakan aplikasi *android Illuminance – Lux Light Meter*.



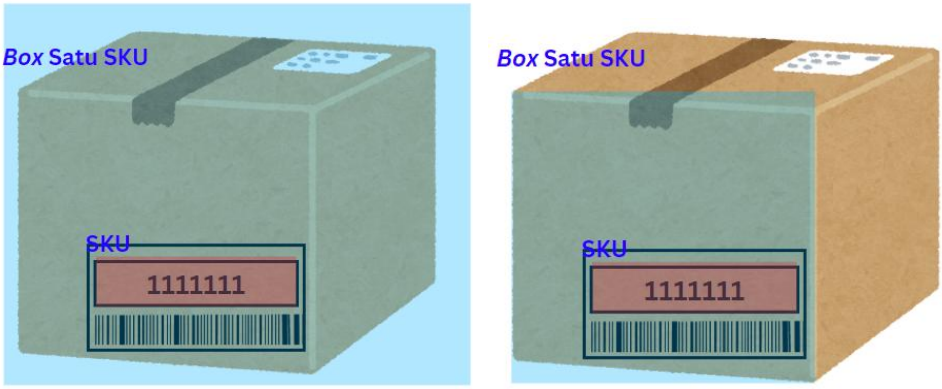
(a) (b)

Gambar 4.1 Pengukuran Pencahayaan; (a) Pengumpulan d*ataset* pada kondisi gelap, (b) Pengumpulan d*ataset* pada kondisi terang

Setelah *dataset* dikumpulkan dalam *google drive*, peneliti melakukan proses *review* untuk memperhatikan kualitas *dataset* sesuai permintaan. Apabila terdapat data yang tidak variatif pada sudut pengambilan gambar dan desain, maka data akan dihapus dan meminta bantuan tim produksi untuk menambah beberapa data sesuai kebutuhan.

Setelah melalui proses *review* dan *dataset* sudah cukup variatif, maka proses selanjutnya adalah mengunduh data dan mengunggahnya untuk disiapkan di komputer A yang kemudian dilakukan anotasi data. Anotasi data dilakukan menggunakan bantuan *software* aplikasi *Computer Vision Annotation Tools* (CVAT) yang terdapat pada komputer A. Proses anotasi *dataset* dibantu oleh tim *outsource* untuk mempercepat proses anotasi. Tim *outsource* merupakan pihak eksternal yang dipekerjakan secara profesional secara khusus untuk tugas tertentu maupun *project* tertentu dengan tujuan mengoptimalkan efisiensi profesional dan mengurangi biaya. Setelah anotasi data selesai dilakukan, akan kembali dilakukan

review untuk perbaikan agar tidak terdapat *noise* pada anotasi data. Anotasi yang dianggap memiliki *noise* adalah anotasi yang melewati sisi samping dari *box* maupun SKU dan *flag* seperti yang tertera pada gambar 4.1.



(a) (b)

Gambar 4.2 Ilustrasi Anotasi *dataset*; (a) *Noise* pada anotasi *box*, (b) Anotasi tidak *noise*

4.3 *Training* Model

Pada proses *training* model, diperlukan *dataset train* dan *dataset test* yang sudah dipisahkan. Untuk melakukan *train test split*, digunakan *function train_test_split* milik *library* SKLearn dengan proporsi 80:20 yang berarti terdapat 1580 data untuk *training* dan 396 data untuk *test* atau *validation*. *Training* model dilakukan menggunakan komputer A dikarenakan komputer A memiliki *storage* yang lebih besar dengan RAM dan GPU yang lebih besar dibandingkan komputer B, sehingga sesuai untuk kebutuhan proses *training*. Proses *training* dilakukan sebanyak masing-masing lima kali, untuk model YOLOv8n dan YOLOv11n untuk mendapatkan variasi hasil guna memperkaya data pada penelitian ini. Proses *training* model menggunakan *hyperparameter* yang sama antara YOLOv8n dan YOLOv11n seperti pada tabel 3.2.

Hyperparameter yang digunakan pada penelitian kali ini telah disesuaikan untuk mendapatkan hasil yang maksimal. Parameter *imgsz* dijadikan sebesar 640 guna mengurangi jumlah pixel dari gambar asli, tetapi tidak menjadikannya terlalu rendah. Pada *batch* digunakan *batch size* sebesar 64 untuk membagi *dataset* menjadi 64 data tiap *epoch* agar proses *training* terbagi dan menjadi lebih ringan

untuk mendapatkan gradien hasil yang stabil. Parameter *pretrain* diatur menjadi *True* untuk menjadikannya *transfer learning* model original dengan *dataset* dari COCO milik *ultralytics*. Parameter *learning rate* diatur 0.01 dan 0.1 agar *weight* dari model terus diperbaiki. Parameter *weight* dan *dropout* diatur menjadi 0.0005 dan 0.2 untuk mencegah terjadinya *overfit* pada model. *Preprocessing* tentunya juga dibutuhkan untuk menambah variasi pada data, sehingga diterapkan beberapa parameter seperti *flip, scale, translate, hsv*. Parameter *patience* diatur menjadi 300 *epochs*, sehingga setelah 300 *epochs* tidak terdapat perubahan pada mAP, maka proses *training* akan dihentikan untuk mempercepat proses *training* model.

4.3.1 Hasil *Training* Model

Training model untuk YOLOv8n dan YOLOv11n dilakukan sebanyak lima kali untuk memperkaya data dalam penelitian ini. Kedua model diberlakukan *training* dengan pengatruan *hyperparameter* yang sama dan *dataset* yang sama.

4.3.1.1 Hasil *Training* YOLOv8n

Tabel 4.1 Hasil *Training* YOLOv8n

No	mAP(50%)	*Precision*	*Recall*	*Total epochs*
1.	95.67	94.58	92.65	1033
2.	95.65	94.52	92.65	1033
3.	95.65	94.59	92.65	1033
4.	95.65	94.58	92.65	1033
5.	95.67	94.58	92.65	1033

Pada tabel 4.1 menunjukkan hasil *training* model YOLOv8n sebanyak lima kali yang memuat nilai dari mAP(50%), *precision*, *recall*, dan *total epochs*. Hasil *training* yang menggunakan 1580 total data ini memberikan performa yang cukup konsisten pada setiap kali melakukan *training*. Nilai mAP(50%) mendapatkan rentang nilai antara 95.65% dan 95.67%, dengan *precision* berada pada rentang 94.52% hingga 94.59% dan nilai *recall* yang konsisten pada 92.65%. Melalui data akurasi pada tabel 4.1 dapat disimpulkan bahwa model YOLOv8n dapat melakukan *task object detection* dengan baik untuk setiap *dataset train* yang diberikan.

Seluruh repetisi *training* untuk YOLOv8n berakhir pada *epoch* 1033 yang menandakan bahwa tidak terdapat perubahan hasil setelah 300 *epoch* hingga diakhiri pada *epoch* yang ke 1033. Perbedaan hasil pada tiap repetisi *training* yang tidak signifikan ini dapat diartikan bahwa YOLOv8n memiliki kemampuan yang baik sehingga tetap konsisten pada tiap repetisinya.

4.3.1.2 Hasil *Training* YOLOv11n

Tabel 4.2 Hasil *Training* YOLOv11n

No	mAP(50%)	*Precision*	*Recall*	*Total epochs*
1.	96.83	95.35	93.91	1074
2.	96.83	95.35	93.91	1074
3.	96.83	95.35	93.91	1074
4.	96.83	95.35	93.91	1074
5.	96.83	95.35	93.91	1074

Pada tabel 4.2 menunjukkan hasil *training* model YOLOv11n sebanyak lima kali yang memuat nilai dari mAP(50%), *precision*, *recall*, dan *total epochs*. Hasil *training* dengan 1580 total data ini memberikan performa yang sangat konsisten pada setiap kali melakukan *training*. Nilai mAP(50%) mendapatkan nilai 96.83%, dengan *precision* berada pada 95.35, dan nilai *recall* yang konsisten pada 93.91%. Melalui data akurasi pada tabel 4.2 dapat disimpulkan bahwa model YOLOv11n dapat melakukan *task object detection* dengan baik untuk setiap *dataset train* yang diberikan.

Seluruh repetisi *training* untuk YOLOv11n berakhir pada *epoch* 1074 yang menandakan bahwa tidak terdapat perubahan hasil setelah 300 *epoch* hingga diakhiri pada *epoch* yang ke 1074. Setelah lima kali repetisi *training* pada model YOLOv11n dengan *dataset* yang sama, didapatkan hasil yang sama untuk setiap repetisinya, sehingga dapat diartikan bahwa YOLOv11n sangat stabil dan konsisten untuk tiap repetisinya.

Dalam perbandingan melalui *accuracy* yang dihasilkan dari YOLOv8n dan YOLOv11n dapat dilihat bahwa YOLOv11n memiliki selisih 1.21%. Diketahui pada penelitian terdahulu bahwa YOLOv8n telah mengalami kenaikan *accuracy*

dibandingkan dengan pendahulunya seperti YOLOv6n yang sebesar 89.8% menjadi 90.6% atau kenaikan sebesar 0.89% (Liu, dan Li, 2024). Dari sini dapat dilihat bahwa YOLO selalu konsisten memperbarui model miliknya dari segi arsitekturnya, sehingga membawa perubahan dan peningkatan pada setiap versinya, sesuai dengan temuan pada hasil *training* YOLOv11n pada penelitian ini.

4.4 *Export* Model

Pada proses *export* model, atau mengoptimalkan model menjadi TensorRT dan OpenVINO pada penelitian ini diberlakukan pada setiap model yang sudah selesai proses *training*. Seluruh model yang ada akan dioptimalkan dengan cara *export* model menjadi format *engine* untuk TensorRT dan *openvino* yang memiliki XML untuk *network topology*, BIN yang menjelaskan *weights* dan *biases*, *Mapping* untuk OpenVINO. Proses *export* model YOLO menjadi format *engine* untuk pada TensorRT dilakukan menggunakan komputer B yang memiliki GPU dari NVIDIA, sehingga sesuai dengan teknik optimisasi TensorRT yang secara spesifik menggunakan GPU milik NVIDIA. Sedangkan untuk *export* model menjadi format *openvino*, dilakukan menggunakan komputer A yang memiliki *processor* Intel XEON, sehingga sesuai dengan teknik optimasi dari OpenVINO yang menggunakan perangkat CPU milik Intel.

Dalam melakukan optimisasi model melalui *export* ini, terdapat parameter yang dapat disesuaikan untuk masing-masing metode, baik dari TensorRT maupun dari OpenVINO. Tujuan dari menyesuaikan parameter ini adalah mengaktifkan dan menyesuaikan teknik-teknik optimisasi model yang dimiliki oleh masing-masing *inference framework*. Seperti contohnya pada TensorRT, terdapat *format* untuk menentukan format *inference framework*, INT8 untuk mengaktifkan *quantization* menjadi INT8, *half* untuk mengaktifkan *half-precision floating-point* menjadi FP16, *simplify* untuk menghilangkan *nodes* yang tidak digunakan, *workspace* untuk mengalokasikan jumlah maksimal dari *memory* yang akan digunakan untuk *optimization* dan *inference*, *batch* untuk memberikan *batch* secara spesifik. Pada OpenVINO, terdapat parameter yang lebih sederhana untuk mengaktifkan teknik optimisasinya, yaitu *format* untuk menentukan *inference framework*, *imgsz* untuk mengaktifkan *resize* pada *input*, *half* untuk mengaktifkan *quantization* menjadi FP16, INT8 untuk mengaktifkan *quantization* menjadi INT8, *batch* untuk

memberikan *batch* secara spesifik, *dynamic* untuk mengaktifkan kemampuan *inference* untuk berbagai ukuran *input*. Pada penelitian ini, untuk *export* menjadi TensorRT, diaktifkan *format* menjadi *engine*, *half* menjadi *True*, *simplify* menjadi *True*, dan *workspace* menjadi 50 serta menggunakan *device* 0. Pada *export* model menjadi OpenVINO, diberikan parameter *imgsz* menjadi 640, *half* menjadi *True*, dan *dynamic* menjadi *True*. Tabel 4.3 hingga tabel 4.7 merupakan tabel rekapitulasi indikator perbandingan dari lima kali *export* model terhadap lima model yang telah diberlakukan *training* sebelumnya.

Tabel 4.3 Hasil *Export* Model Replikasi 1

Indikator	**TensorRT**		**OpenVINO**	
	YOLOv8	**YOLOv11**	**YOLOv8**	**YOLOv11**
Inference speed (s)	0.0059	0.0066	0.0173	0.0168
Memory consumption (MB)	4175.36	4176.77	2428.08	3178.16
Model size (MB)	8.3	7.8 MB	6.4	5.6
Loading time (s)	0.0006589	0.0009272	0.002273	0.0032367
Accuracy (%)	95.27	96.32	95.23	96.22

Hasil *export* model pada replikasi pertama untuk melakukan perbandingan kinerja antara TensorRT dan OpenVINO pada YOLOv8n dan YOLOv11n pada beberapa indikator pengujian disajikan pada tabel 4.3. Seperti contohnya pada *inference speed*, TensorRT memiliki kecepatan yang lebih baik dibandingkan dengan OpenVINO pada YOLOv8n dan YOLOv11n, pada indikator *memory consumption* OpenVINO memiliki kemampuan lebih baik karena membutuhkan RAM yang lebih sedikit ketika melakukan *inference*, pada *model size* untuk replikasi pertama ini OpenVINO lebih unggul karena ukuran model yang lebih kecil dan ringan serta adanya penurunan *model size* pada model YOLOv11n. Pada

indikator *loading time* terlihat bahwa TensorRT memiliki kecepatan *load* model lebih cepat, sedangkan pada *accuracy* tidak terlihat perbedaan signifikan pada kedua *inference framework*, tetapi terlihat adanya sedikit kenaikan *accuracy* jika menggunakan model YOLOv11n.

Tabel 4.4 Hasil *Export* Model Replikasi 2

Indikator	**TensorRT**		**OpenVINO**	
	YOLOv8	**YOLOv11**	**YOLOv8**	**YOLOv11**
Inference speed (s)	0.0057	0.0068	0.0171	0.0166
Memory consumption (MB)	4081.616	4077.600	2674.860	3172.896
Model size (MB)	8.3	7.9	6.4	5.6
Loading time (s)	0.00069713	0.0007345	0.0035126	0.0036258
Accuracy (%)	95.26	96.3	95.23	96.22

Hasil *export* model pada replikasi kedua untuk melakukan perbandingan kinerja antara TensorRT dan OpenVINO pada YOLOv8n dan YOLOv11n pada beberapa indikator pengujian disajikan pada tabel 4.4. Pada indikator *inference speed* terlihat TensorRT memiliki kecepatan yang lebih baik dibandingkan dengan OpenVINO pada YOLOv8n dan YOLOv11n, pada indikator *memory consumption* OpenVINO memiliki kemampuan lebih baik karena membutuhkan RAM yang lebih sedikit ketika melakukan *inference*, pada *model size* untuk replikasi kedua OpenVINO masih lebih unggul karena ukuran model yang lebih kecil dan ringan serta adanya penurunan *model size* pada model YOLOv11n sama seperti replikasi sebelumnya. Pada indikator *loading time* terlihat bahwa TensorRT memiliki kecepatan *load* model lebih cepat, sedangkan pada *accuracy* tidak terlihat

perbedaan signifikan pada kedua *inference framework*, tetapi tetap terlihat adanya kenaikan *accuracy* jika menggunakan model YOLOv11n.

Tabel 4.5 Hasil *Export* Model Replikasi 3

Indikator	**TensorRT**		**OpenVINO**	
	YOLOv8	**YOLOv11**	**YOLOv8**	**YOLOv11**
Inference speed (s)	0.005	0.0067	0.0171	0.0167
Memory consumption (MB)	2096.992	4079.976	2809.352	3248.260
Model size (MB)	8.2	7.7	6.4	5.6
Loading time (s)	0.0006437	0.0006775	0.0027019	0.0030105
Accuracy (%)	95.21	96.17	95.23	96.22

Hasil *export* model pada replikasi ketiga untuk melakukan perbandingan kinerja antara TensorRT dan OpenVINO pada YOLOv8n dan YOLOv11n pada beberapa indikator pengujian disajikan pada tabel 4.5. Pada indikator *inference speed* masih sama seperti dua replikasi sebelumnya, yaitu terlihat TensorRT lebih cepat melakukan *inference* jika dibandingkan dengan OpenVINO pada YOLOv8n dan YOLOv11n. Pada indikator *memory consumption* terlihat perubahan kemampuan pada *inference framework* TensorRT khususnya pada model YOLOv8n yang mengalami penurunan signifikan pada, tetapi tidak terlihat pada model YOLOv11n, sedangkan pada OpenVINO juga tidak terlihat perubahan signifikan jika dibandingkan dengan dua replikasi sebelumnya. Pada *model size* untuk replikasi ketiga OpenVINO masih lebih unggul karena ukuran model yang lebih kecil dan ringan serta adanya penurunan *model size* pada model YOLOv11n

sama seperti replikasi sebelumnya. Pada indikator *loading time* terlihat bahwa TensorRT memiliki kecepatan *load* model lebih cepat, sedangkan pada *accuracy* tidak terlihat perbedaan signifikan pada kedua *inference framework*, tetapi tetap terlihat adanya kenaikan *accuracy* jika menggunakan model YOLOv11n.

Tabel 4.6 Hasil *Export* Model Replikasi 4

Indikator	**TensorRT**		**OpenVINO**	
	YOLOv8	**YOLOv11**	**YOLOv8**	**YOLOv11**
Inference speed (s)	0.0055	0.0068	0.0173	0.0165
Memory consumption (MB)	2102.776	4075.564	2947.048	3293.976
Model size (MB)	8.4	7.7	6.4	5.6
Loading time (s)	0.0007231	0.0006728	0.0028142	0.0026895
Accuracy (%)	95.27	96.30	95.23	96.22

Hasil *export* model pada replikasi keempat untuk melakukan perbandingan kinerja antara TensorRT dan OpenVINO pada YOLOv8n dan YOLOv11n pada beberapa indikator pengujian disajikan pada tabel 4.3. Pada indikator *inference speed* masih sama seperti tiga replikasi sebelumnya, yaitu terlihat TensorRT lebih cepat melakukan *inference* jika dibandingkan dengan OpenVINO pada model YOLOv8n dan YOLOv11n. Penurunan *memory consumption* pada YOLOv8n dengan *inference framework* TensorRT kembali terlihat di replikasi keempat ini, sedangkan pada model YOLOv11n dan kedua model dengan OpenVINO tidak terlihat adanya perubahan signifikan. Pada *model size* untuk replikasi keempat OpenVINO kembali unggul dengan ukuran model yang lebih kecil dan ringan serta adanya penurunan *model size* pada model YOLOv11n sama seperti replikasi sebelumnya. Pada indikator *loading time* terlihat bahwa TensorRT lebih cepat

dalam kemampuan *load* model, sedangkan pada *accuracy* masih tidak terlihat perbedaan signifikan pada kedua *inference framework*, tetapi tetap terlihat adanya kenaikan *accuracy* jika menggunakan model YOLOv11n.

Tabel 4.7 Hasil *Export* Model Replikasi 5

Indikator	**TensorRT**		**OpenVINO**	
	YOLOv8	**YOLOv11**	**YOLOv8**	**YOLOv11**
Inference speed (s)	0.0051	0.0063	0.0169	0.0166
Memory consumption (MB)	2098.904	4073.968	3026.500	3352.200
Model size (MB)	8.0	7.7	6.4	5.6
Loading time (s)	0.0006904	0.0006697	0.0030753	0.0032799
Accuracy (%)	95.24	96.31	95.23	96.22

Hasil *export* model pada replikasi kelima untuk melakukan perbandingan kinerja antara TensorRT dan OpenVINO pada YOLOv8n dan YOLOv11n pada beberapa indikator pengujian disajikan pada tabel 4.3. Pada indikator *inference speed* masih sama seperti empat replikasi sebelumnya, yaitu terlihat TensorRT lebih cepat jika dibandingkan dengan OpenVINO pada model YOLOv8n dan YOLOv11n. Penurunan *memory consumption* pada YOLOv8n dengan *inference framework* TensorRT kembali terjadi di replikasi keempat ini yang menandakan bahwa replikasi *export* model membawa dampak, sedangkan pada model YOLOv11n dan kedua model dengan OpenVINO masih tidak terlihat adanya perubahan signifikan. Pada *model size* untuk replikasi kelima OpenVINO kembali unggul dengan ukuran model yang lebih kecil dan ringan sekitar 1 MB dari TensorRT, serta adanya penurunan *model size* pada model YOLOv11n sama seperti

replikasi sebelumnya. Pada indikator *loading time* terlihat bahwa TensorRT masih unggul dalam kemampuan *load* model, sedangkan pada *accuracy* masih tidak terlihat perbedaan signifikan setelah lima kali replikasi pada kedua *inference framework*, tetapi tetap terlihat adanya kenaikan *accuracy* jika menggunakan model YOLOv11n dengan selisih sekitar 1%.

Melalui rekapitulasi hasil *export* model dengan replikasi lima kali, belum dapat dilihat hasil signifikan pada kedua *inference framework* dengan variasi model YOLOv8n dan YOLOv11n. Maka dari itu diperlukan kajian lebih mendalam pada hasil *export* model dengan replikasi lima kali ini.

4.5 Pembahasan Perbandingan Kinerja

Pada sub bab ini akan membahas lebih mendalam mengenai hasil dari *export* model YOLOv8n dan YOLOv11n menjadi *inference framework* TensorRT dan OpenVINO. Pada perbandingan kinerja ini akan dilakukan tiga metode pengujian agar didapatkan hasil dengan pembahasan yang lebih mendalam. Ketiga metode pengujian adalah melalui rata-rata, kemudian melalui uji T *independent* untuk melihat perbedaan rata-rata kedua kelompok signifikan atau tidak, dan uji *two-way* ANOVA untuk mencari tahu relasi atau interaksi pada kombinasi tertentu antara model dengan *inference framework*.

4.5.1 Perbandingan Rata-rata

Perbandingan rata-rata atau *mean* dapat dilakukan dengan cukup sederhana melalui perhitungan rata-rata dari indikator perbandingan terhadap model dan *inference framework* yang telah dilakukan dengan replikasi sebanyak lima kali.

4.5.1.1 *Inference Speed*

Tabel 4.8 Perbandingan Rata-rata *Inference Speed*

Model	TensorRT	OpenVINO
YOLOv8n	0.00544 s	0.01714 s
YOLOv11n	0.00664 s	0.01664 s

Tabel 4.8 menyajikan data rata-rata *inference speed* untuk TensorRT dan OpenVINO saat menggunakan model YOLOv8n dan YOLOv11n. Dalam penggunaan *inference framework* TensorRT, dapat dilihat bahwa rata-rata *inference*

speed untuk model YOLOv8n adalah 0.00544 detik dan untuk YOLOv11n adalah 0.00664 detik dengan selisih antara kedua *inference speed* pada kedua model adalah sebesar 0.0012 detik. Pada *inference framework* OpenVINO, terlihat bahwa menggunakan model YOLOv8n adalah 0.0171 detik, sedangkan saat menggunakan YOLOv11n *inference speed* mengalami peningkatan menjadi 0.0166 detik dengan selisih sebesar 0.0005 detik.

4.5.1.2 *Memory Consumption*

Tabel 4.9 Perbandingan Rata-rata *Memory Consumption*

Model	TensorRT	OpenVINO
YOLOv8n	2891.557 MB	2765.787 MB
YOLOv11n	4077.197 MB	3234.202 MB

Tabel 4.9 menyajikan perbandingan rata-rata dari *memory consumption* untuk TensorRT dan OpenVINO pada saat melakukan *inference* menggunakan model YOLOv8n dan YOLOv11n. Dalam penggunaan *inference framework* TensorRT, dapat dilihat bahwa rata-rata *memory consumption* untuk model YOLOv8n adalah 2891.557 MB dan untuk YOLOv11n adalah 4077.197 MB yang menunjukkan bahwa penggunaan TensorRT pada model YOLOv8n. Pada OpenVINO didapatkan *memory consumption* sebesar 2765.787 MB untuk YOLOv8n dan 3234.202 MB untuk YOLOv11n.

4.5.1.3 *Model Size*

Tabel 4.10 Perbandingan Rata-rata *Model Size*

Model	TensorRT	OpenVINO
YOLOv8n	8.24 MB	6.4 MB
YOLOv11n	7.76 MB	5.6 MB

Tabel 4.10 menyajikan perbandingan rata-rata dari *model size* untuk TensorRT dan OpenVINO saat menggunakan model YOLOv8n dan YOLOv11n. Dalam penggunaan *inference framework* TensorRT, dapat dilihat bahwa rata-rata *model size* untuk model YOLOv8n adalah 8.24 MB dan untuk YOLOv11n adalah 7.76 MB yang memiliki selisih sebesar 0.48 MB. Pada *inference framework*

OpenVINO, terlihat bahwa rata-rata *model size* untuk YOLOv8n adalah 6.4 MB dan pada YOLOv11n adalah 5.6 MB yang memiliki selisih 0.8 MB.

4.5.1.4 *Load Time*

Tabel 4.11 Perbandingan Rata-rata *Load Time*

Model	TensorRT	OpenVINO
YOLOv8n	0.000682646 s	0.0028754 s
YOLOv11n	0.00073634 s	0.00316848 s

Tabel 4.11 menyajikan perbandingan rata-rata dari *load time* untuk TensorRT dan OpenVINO saat menggunakan model YOLOv8n dan YOLOv11n. Dalam penggunaan *inference framework* TensorRT, dapat dilihat bahwa rata-rata *load time* untuk model YOLOv8n adalah 0.00068 detik dan untuk YOLOv11n adalah 0.00073 detik. Pada *inference framework* OpenVINO, terlihat bahwa YOLOv8n membutuhkan waktu 0.00287 detik, sedangkan untuk YOLOv11n membutuhkan waktu 0.00316 detik untuk melakukan *load model*.

4.5.1.5 *Accuracy*

Tabel 4.12 Perbandingan Rata-rata *Accuracy*

Model	TensorRT	OpenVINO
YOLOv8n	95.25 %	95.23 %
YOLOv11n	96.28 %	96.22 %

Tabel 4.12 menyajikan perbandingan rata-rata dari *accuracy* dengan mAP50 untuk TensorRT dan OpenVINO saat menggunakan model YOLOv8n dan YOLOv11n. Dapat dilihat bahwa rata-rata *accuracy* saat menggunakan TensorRT untuk model YOLOv8n adalah 95.25% dan untuk YOLOv11n adalah 96.28% dengan selisih sebesar 1.03%. Pada *inference framework* OpenVINO terlihat bahwa YOLOv8n memberikan *accuracy* sebesar 95.23% dan YOLOv11n menghasilkan *accuracy* sebesar 96.22% dengan selisih *accuracy* sebesar 0.99%.

4.5.2 Uji T

Data yang dimiliki pada penelitian ini merupakan dua set data, maka *one-sample T-Test* tidak sesuai dengan kondisi data. Setelah itu, pilihan uji T hanya pada

independent T-Test atau *paired T-Test* dilihat dari kondisi relasi pada data. Data pertama dengan data kedua tidak memiliki relasi yang saling berkaitan, karena variabel pengamatan yang berbeda, yaitu *inference framework* TensorRT dan OpenVINO yang memiliki arsitektur masing-masing yang khas dan tidak berkaitan. Maka jenis uji T yang sesuai untuk pengujian perbedaan signifikan antara kedua data dari *inference framework* adalah *independent T-Test*.

Uji *levene* yang digunakan pada penelitian ini adalah menggunakan pemrograman python dengan *library scipy*, yaitu *levene*. Parameter yang digunakan pada uji *levene* pada penelitian ini adalah kedua data dan *function* pengujiannya yaitu *mean* atau rata-rata dari data. Dengan kembalian nilai dari fungsi *levene* ini adalah hasil uji statistiknya dan *p-value* untuk menentukan tolak H_0 yang berarti tidak ada homogenitas pada data atau gagal tolak H_0 yang berarti terdapat homogenitas pada data. Penentuan tolak H_0 atau gagal tolak H_0 dapat dilihat dari membandingkan nilai *alpha* dengan nilai *p-value*, jika *p-value* < *alpha* maka akan tolak H_0 yang berarti tidak ada homogenitas pada data dan jika sebaliknya maka gagak tolak H_0 yang berarti terdapat homogenitas pada data.

4.5.2.1 Hasil Uji *Levene T-Test*

Uji *levene* pada penelitian ini dilakukan dengan tujuan untuk menentukan metode yang tepat dalam melakukan *T-Test* yang akan dilakukan berikutnya. Pada penelitian ini, uji *levene* dilakukan dengan menggunakan bahasa pemrograman python dengan bantuan *library scipy* dengan fungsi *levene()*. Fungsi *levene()* menerima dua buah parameter yang terdiri atas set data yang akan diuji dan metode pengujiannya atau *center*. Parameter metode pengujian atau *center* pada uji *levene* dengan *scipy* bertujuan untuk menghitung pusat distribusi data, umumnya akan digunakan *mean* atau rata-rata dari data untuk mendapatkan pusat distribusi data. Pada penelitian ini, dua set data yang digunakan adalah model YOLOv8n dan YOLOv11n ketika menggunakan *inference framework* TensorRT dan OpenVINO, dengan masing-masing indikator perbandingan dan *center* menggunakan *mean*. Pengujian masing-masing dilakukan dengan melihat homogenitas antara data indikator perbandingan dengan model tertentu ketika menggunakan TensorRT dan OpenVINO, sehingga total dilakukan sepuluh kali pengujian dengan rincian lima kali untuk YOLOv8n dengan lima indikator perbandingan dan YOLOv11n dengan

lima indikator perbandingan. Dalam uji *levene* untuk penelitian ini, ditetapkan nilai *threshold alpha* adalah 0.05 dengan hipotesis nol adalah tidak ada perbedaan signifikan antara dua set data atau homogen.

Tabel 4.13 Hasil Uji *Levene T-Test*

Model	***Category***	***p-value***	**Status**	***Levene***
YOLOv8	*Inference speed*	0.0557	Gagal Tolak H0	*True*
YOLOv8	*Memory consumption*	0.0002	Tolak H0	*False*
YOLOv8	*Model size*	0.0191	Tolak H0	*False*
YOLOv8	*Load time*	0.0314	Tolak H0	*False*
YOLOv8	mAP50	0.0065	Tolak H0	*False*
YOLOv11	*Inference speed*	0.3101	Gagal Tolak H0	*True*
YOLOv11	*Memory consumption*	0.0079	Tolak H0	*False*
YOLOv11	*Model size*	0.0033	Tolak H0	*False*
YOLOv11	*Load time*	0.0931	Gagal Tolak H0	*True*
YOLOv11	mAP50	0.0315	Tolak H0	*False*

Pada kedua model, untuk indikator *inference speed* didapatkan bahwa *p-value* bernilai lebih dari *threshold alpha*, maka tidak ditemukan adanya perbedaan *variance* secara signifikan atau dapat disebut data homogen dan *levene* menjadi bernilai *True* untuk parameter *T-Test*. Berikutnya, pada indikator *memory consumption* ditemukan bahwa *p-value* dari pengujian ini kurang dari *threshold alpha* yang berarti ditemukan perbedaan data signifikan atau dapat disebut data tidak homogen dan *levene* menjadi bernilai *False*. Untuk indikator *model size* kembali ditemukan bahwa *p-value* kurang dari *threshold alpha* yang berarti perbedaan pada data TensorRT dan OpenVINO signifikan atau dapat diartikan bahwa data bukan data homogen untuk model YOLOv8n dan YOLOv11n. Pada indikator perbandingan *load time*, ditemukan perbedaan hasil uji *levene* di model YOLOv8n dan YOLOv11n. Untuk model YOLOv8n, *p-value* dari uji *levene* kurang dari *threshold alpha* yang berarti tidak ditemukan perbedaan signifikan pada *load time* menggunakan TensorRT ataupun OpenVINO sehingga dapat disebut data

homogen, sedangkan pada YOLOv11n, ditemukan bahwa *p-value* lebih dari *threshold alpha* yang berarti terdapat perbedaan signifikan pada *load time* untuk *inference framework* TensorRT dan OpenVINO sehingga data disebut tidak homogen. Indikator *accuracy* pada kedua model dengan data *inference framework* TensorRT dan OpenVINO ditemukan bahwa *p-value* hasil uji *levene* kurang dari *threshold alpha*, sehingga ditemukan perbedaan signifikan pada kedua data atau data tidak bersifat homogen.

Hasil uji *levene* yang menunjukkan adanya sifat homogenitas pada data, memberikan arti bahwa keseluruhan data cenderung memiliki nilai yang sama bahkan mirip, sehingga dapat dikatakan bahwa data lebih stabil. Sebagai contoh pada indikator *load time*, ditemukan bahwa pengujian *levene* untuk data YOLOv8n dengan TensorRT dan OpenVINO ditemukan sifat tidak homogen pada data, dan pada YOLOv11n ditemukan sifat homogen pada data *inference framework* TensorRT dan OpenVINO, yang berarti model YOLOv11n dapat dengan mudah dimuat ketika menggunakan *inference framework* TensorRT maupun OpenVINO oleh komputer. Pada hasil uji *levene*, ditambahkan kolom *levene* yang terdiri dari *True* dan *False* yang dapat digunakan sebagai parameter untuk *T-Test*. Jika *levene* bernilai *True*, maka data disebut homogen dan jika *False*, maka data disebut tidak homogen.

Melalui uji *levene* pada setiap *inference framework* dengan penggunaan model yang berbeda, dapat diambil kesimpulan dini mengenai kedua model dengan *inference framework* yang berbeda. Model YOLOv8n dan YOLOv11n menunjukkan *inference speed* yang memiliki *variance* homogen, sehingga mencerminkan konsistensi performa pada dua *inference framework*. Pada data *memory consumption*, model YOLOv11n cenderung memiliki *variance* yang lebih rendah pada dua *inference framework* dibandingkan dengan model YOLOv8n, sehingga YOLOv11n dapat diandalkan pada berbagai kondisi. Namun, pada *model size*, ditemukan *variance inference framework* yang lebih besar pada YOLOv11n, sehingga perlu mempertimbangkan *storage* untuk penyimpanan. Di sisi lain, YOLOv11n lebih stabil dalam melakukan *load time*, baik ketika menggunakan TensorRT maupun OpenVINO, jika dibandingkan dengan kemampuan *load time* dari YOLOv8n. Pada indikator *accuracy* melalui uji *levene*, ditemukan bahwa

variance pada YOLOv11n lebih kecil ketika menggunakan TensorRT maupun OpenVINO, yang berarti model YOLOv11n cenderung stabil.

4.5.2.2 Hasil Uji T

Pada penelitian ini, pengujian *T*-Test dilakukan dengan bantuan program python dengan *library scipy*, yang memiliki sebuah fungsi *ttest_ind()*. Fungsi *ttst_ind()* ini menerima dua parameter, yang pertama adalah set data untuk pengujian dan penanda sifat data homogen. Set data yang dijadikan parameter pada penelitian ini adalah *inference framework* TensorRT dengan OpenVINO dan menggunakan *category* atau indikator perbandingan sebagai variabelnya, sedangkan *True* atau *False* sebagai penanda sifat homogen pada data yang disiapkan melalui hasil uji *levene*. Pada tabel 4.14 disajikan hasil dari *T-Test* yang membandingkan performa dari *inference framework* TensorRT milik NVIDIA dengan OpenVINO milik Intel dengan pemodelan YOLOv8n. *T-Test* pada penelitian kali ini dilakukan terhadap lima indikator perbandingan yang telah disebutkan sebelumnya, sehingga total hasil dari *T-Test* ini adalah lima baris data hasil pengujian terhadap lima indikator perbandingan. Pada *T-Test* penelitian ini, *threshold alpha* ditetapkan menjadi 0.05 dengan hipotesis nol bahwa tidak terdapat perbedaan pada rata-rata pada kedua *inference framework* dengan masing-masing kelompok data indikator perbandingan.

Tabel 4.14 Hasil *T-Test* YOLOv8n

Model	***Category***	***p-value***	**Status**	***Mean* TensorRT**	***Mean* OpenVINO**
YOLO v8n	*Inference speed*	0.0000	Tolak H0	0.00544 s	0.01714 s
YOLO v8n	*Memory consumption*	0.8072	Gagal Tolak H0	2891.557 MB	2765.787 MB
YOLO v8n	*Model size*	0.0000	Tolak H0	8.24 MB	6.4 MB
YOLO v8n	*Load time*	0.0000	Tolak H0	0.00068 s	0.00287 s

YOLO v8n	mAP50	0.1175	Gagal Tolak H0	95.25 %	95.23 %

Pada Tabel 4.14 disajikan hasil dari *T-Test* yang membandingkan performa dari *inference framework* TensorRT milik NVIDIA dengan OpenVINO milik Intel dengan pemodelan YOLOv8n. *T-Test* pada penelitian kali ini dilakukan terhadap lima indikator perbandingan yang telah disebutkan sebelumnya, sehingga total hasil dari *T-Test* ini adalah lima baris data hasil pengujian terhadap lima indikator perbandingan. Pada *T-Test* penelitian ini, *threshold alpha* ditetapkan menjadi 0.05 dengan hipotesis nol bahwa tidak terdapat perbedaan pada rata-rata pada kedua *inference framework* dengan masing-masing kelompok data indikator perbandingan.

Pada indikator perbandingan *inference speed* dengan model YOLOv8n didapatkan bahwa *p-value* kurang dari *threshold alpha*, yang berarti terdapat perbedaan signifikan pada rata-rata *inference speed* untuk *inference framework* TensorRT dan OpenVINO, jika melihat nilai rata-ratanya, maka dapat disimpulkan bahwa TensorRT lebih unggul tiga kali lipat dalam *inference speed*. Hasil dari *T-Test* pada kedua *inference framework* untuk indikator *memory consumption* adalah *p-value* lebih dari *threshold alpha*, sehingga dapat disimpulkan bahwa tidak terdapat perbedaan signifikan pada kedua rata-rata jumlah *memory consumption* untuk model YOLOv8n. Pada indikator *model size*, *p-value* untuk hasil *T-Test* adalah kurang dari *threshold alpha*, yang berarti bahwa terdapat perbedaan signifikan antara *model size* YOLOv8n dengan *inference framework* TensorRT dan OpenVINO dengan OpenVINO lebih unggul dalam ukuran yang ringan, sehingga dapat lebih meringankan beban pada *storage* komputer. Pada indikator *load time*, hasil dari *T-Test* menyatakan bahwa nilai *p-value* kurang dari *threshold alpha*, yang menunjukkan bahwa terdapat perbedaan signifikan antara *load time* pada kedua *inference framework* dengan TensorRT yang unggul hampir empat kali lebih cepat, sehingga TensorRT dapat memberi keuntungan untuk kebutuhan kecepatan dalam tahap *deployment*. Pada indikator *accuracy* berdasarkan nilai mAP50 dari kedua *inference framework*, hasil *T-Test* menunjukkan bahwa *p-value* lebih dari *threshold alpha*, sehingga dapat disimpulkan bahwa tidak terdapat perbedaan signifikan pada

accuracy kedua *inference framework*, meskipun terlihat TensorRT unggul 0.02% dari OpenVINO.

Secara keseluruhan untuk YOLOv8n, *inference framework* TensorRT lebih unggul pada kategori *inference speed* sebesar tiga kali lipat daripada OpenVINO, sehingga sangat sesuai untuk penggunaan yang sangat mempertimbangkan kecepatan dalam melakukan *inference* atau pada kasus *real-time processing*. Untuk *memory consumption* pada kedua *inference framework* tidak begitu signifikan, sehingga keduanya tidak menjadi masalah dalam hal batasan perangkat keras secara khusus pada RAM. Berikutnya pada kategori *model size*, dapat dilihat bahwa OpenVINO memiliki kemampuan yang sedikit lebih unggul daripada TensorRT karena mampu melakukan konversi model dengan optimal sehingga membuat *model size* lebih kecil dan membuat OpenVINO sesuai untuk kebutuhan perangkat yang terbatas pada ukuran *storage*. Pada kategori *load time*, TensorRT ternyata lebih unggul 0.00219 detik daripada menggunakan OpenVINO, sehingga kembali mendukung kebutuhan kecepatan *inference* daripada menggunakan OpenVINO. Namun, pada indikator *accuracy* dengan mAP50 tidak ditemukan perbedaan yang signifikan dari kedua *inference framework* dalam penggunaannya pada model YOLOv8n.

Tabel 4.15 Hasil *T-Test* YOLOv11n

Model	***Category***	***p-value***	**Status**	***Mean* TensorRT**	***Mean* OpenVINO**
YOLO v11n	*Inference speed*	0.0000	Tolak H0	0.00664 s	0.01664 s
YOLO v11n	*Memory consumption*	0.0000	Tolak H0	4077.197 MB	3234.202 MB
YOLO v11n	*Model size*	0.0000	Tolak H0	7.76 MB	5.6 MB
YOLO v11n	*Load time*	0.0000	Tolak H0	0.00073 s	0.00316 s

YOLO v11n	mAP50	0.0626	Gagal Tolak H0	96.28 %	96.22 %

Tabel 4.15 menunjukkan hasil dari *T-Test* yang membandingkan performa dari *inference framework* TensorRT milik NVIDIA dengan OpenVINO milik Intel dengan pemodelan YOLOv11n. *T-Test* kali kali ini dilakukan terhadap lima indikator perbandingan sama seperti *T-Test* untuk YOLOv8n dengan total hasil dari *T-Test* ini adalah lima baris data hasil pengujian terhadap lima indikator perbandingan. Pada *T-Test* YOLOv11n ini, *threshold alpha* ditetapkan menjadi 0.05 dengan hipotesis nol bahwa tidak terdapat perbedaan pada rata-rata pada kedua *inference framework* dengan masing-masing kelompok data indikator perbandingan. Pada indikator perbandingan *inference speed* dengan model YOLOv11n, didapatkan bahwa *p-value* kurang dari *threshold alpha* yang ditentukan, hal ini berarti bahwa terdapat perbedaan signifikan pada rata-rata *inference speed* untuk *inference framework* TensorRT dan OpenVINO. Berdasarkan nilai rata-rata yang diperoleh, dapat disimpulkan bahwa TensorRT lebih unggul hampir sebesar tiga kali lipat dalam *inference speed* jika dibandingkan OpenVINO, menjadikannya lebih ideal untuk kebutuhan yang memprioritaskan *real-time processing*. Pada indikator *memory consumption*, melalui hasil *T-Test*, ditemukan bahwa *p-value* kurang dari *threshold alpha*, yang menunjukkan adanya perbedaan signifikan pada rata-rata *memory consumption* antara TensorRT dan OpenVINO. OpenVINO menunjukkan penggunaan *memory* yang lebih baik dengan konsumsi memori yang lebih rendah sekitar 800.000 MB dibandingkan dengan TensorRT, sehingga lebih sesuai untuk kebutuhan tertentu yang memiliki keterbatasan pada *memory*. Pada indikator *model size*, *p-value* hasil *T-Test* juga kurang dari *threshold alpha* yang telah ditentukan, yang berarti terdapat perbedaan signifikan antara *model size* dari YOLOv11n yang dihasilkan oleh TensorRT dan OpenVINO, dengan OpenVINO kembali unggul dengan *model size* yang lebih kecil dibandingkan TensorRT, memberikan keunggulan dalam hal penghematan *storage* dan fleksibilitas dalam *deployment* pada perangkat dengan sumber daya terbatas. Pada indikator *load time*, hasil *T-Test* menunjukkan bahwa *p-value* kurang dari *threshold alpha*, yang mengindikasikan terdapat perbedaan signifikan antara *load*

time model pada kedua *inference framework*. TensorRT terbukti hampir empat kali lebih cepat dibandingkan OpenVINO, memberikan keunggulan yang signifikan untuk kebutuhan *deployment* model yang cepat. Pada indikator *accuracy* berdasarkan mAP50, melalui *T-Test*, didapatkan bahwa *p*-value lebih dari threshold alpha yang menunjukkan tidak terdapat perbedaan yang signifikan pada *accuracy* antara kedua *inference framework*, meskipun TensorRT lebih unggul dengan selisih 0.06% dibandingkan dengan OpenVINO.

Secara keseluruhan untuk YOLOv11n, *inference framework* TensorRT kembali unggul pada kategori *inference speed* daripada menggunakan OpenVINO, sehingga sesuai untuk penggunaan yang sangat mempertimbangkan kecepatan dalam melakukan *inference*. Pada indikator *memory consumption* untuk kedua *inference framework*, terlihat perbedaan yang cukup signifikan dimana OpenVINO unggul karena OpenVINO menggunakan RAM yang lebih sedikit saat melakukan *inference*, sehingga hal ini dapat dijadikan pertimbangan untuk kebutuhan perangkat yang terbatas pada RAM. Selanjutnya, pada kategori *model size*, dapat dilihat bahwa OpenVINO kembali unggul daripada TensorRT karena mampu melakukan konversi model dengan lebih optimal, sehingga membuat *model size* lebih ringan dengan selisih antara ukuran TensorRT dan OpenVINO adalah sekitar 2 MB dan membuat OpenVINO sesuai untuk kebutuhan perangkat yang terbatas pada ukuran *storage*. TensorRT lebih unggul daripada OpenVINO pada indikator *load time*, sehingga masih mendukung dalam segi kebutuhan untuk kecepatan *inference* daripada menggunakan OpenVINO. Namun, pada indikator *accuracy* dengan mAP50 kembali tidak ditemukan perbedaan signifikan antara kedua *inference framework* dalam penggunaannya pada model YOLOv11n.

Dari kedua *T-Test* untuk *inference framework* dalam penggunaan model YOLOv8n dan YOLOv11n, terdapat beberapa kesamaan. Kesimpulan yang dapat diambil dari *T-Test* ini adalah bahwa *inference speed* untuk TensorRT sangat unggul dibandingkan dengan OpenVINO baik ketika menggunakan model YOLOv8n maupun YOLOv11n. Kemudian untuk *memory consumption*, ternyata TensorRT dan OpenVINO tidak memiliki perbedaan signifikan pada YOLOv8n, sedangkan pada model yang lebih kompleks seperti YOLOv11n, OpenVINO unggul dalam penggunaan RAM ketika melakukan *inference*, karena OpenVINO

hanya menggunakan RAM yang lebih sedikit. Berikutnya untuk indikator *model size*, OpenVINO selalu unggul baik saat menggunakan YOLOv8n maupun YOLOv11n. Hal ini mengindikasikan bahwa OpenVINO mampu mengoptimalkan kedua model menjadi lebih sederhana dibandingkan dengan *inference framework* TensorRT. Dari indikator *load time model*, TensorRT menunjukkan keunggulan, karena kemampuannya untuk mengunggah model yang lebih cepat daripada OpenVINO pada model YOLOv8n dan YOLOv11n. Berikutnya dari sisi *accuracy* dengan menggunakan mAP50, ditemukan bahwa TensorRT maupun OpenVINO sama-sama memiliki kemampuan baik dalam melakukan konversi atau *export* model karena mampu mengoptimalkan model YOLOv8n dan YOLOv11n tanpa mengorbankan *accuracy* secara signifikan. Secara keseluruhan model YOLOv11n tetap menunjukkan *accuracy* yang lebih baik saat menggunakan kedua *inference framework*, meskipun OpenVINO sedikit menunjukkan penurunan *accuracy* sekitar 0.04% secara rata-rata. *Accuracy* dari model yang diberlakukan *optimization* dengan *inference framework* OpenVINO ternyata mengalami sedikit penurunan *accuracy* sebesar 0.021% pada YOLOv8n dan 0.062% pada YOLOv11n. Hal ini bertentangan dengan penelitian terdahulu yang dilakukan oleh Biswas, *et al* (2020) yang menyatakan bahwa OpenVINO lebih unggul secara *accuracy*.

4.5.3 Uji ANOVA

Seperti yang dilakukan pada *T-Test*, sebelum melakukan uji ANOVA perlu dilakukan uji *levene* untuk mengetahui *variance* pada data agar pengujian untuk ANOVA dapat sesuai dengan kondisi data sehingga hasil dari pengujian dapat disebut valid.

4.5.3.1 Hasil Uji *Levene* ANOVA

Uji *levene* pada untuk *Two-Way* ANOVA pada penelitian ini dilakukan dengan tujuan untuk menentukan metode yang tepat dalam melakukan *Two-Way* ANOVA yang akan dilakukan berikutnya. Pada penelitian ini, uji *levene* dilakukan dengan menggunakan bahasa pemrograman python dengan bantuan *library scipy* dengan fungsi *levene()*. Fungsi *levene()* menerima dua buah parameter yang terdiri atas set data yang akan diuji dan metode pengujiannya atau *center*. Parameter metode pengujian atau *center* pada uji *levene* dengan *scipy* bertujuan untuk menghitung pusat distribusi data, umumnya akan digunakan *mean* atau rata-rata

dari data untuk mendapatkan pusat distribusi data. Pada penelitian ini, untuk menguji *levene* dengan empat set data, data yang digunakan adalah model YOLOv8n dan YOLOv11n ketika menggunakan *inference framework* TensorRT dan OpenVINO, dengan tiap indikator perbandingan dan *center* menggunakan *mean*. Pengujian masing-masing dilakukan dengan melihat homogenitas antara data indikator perbandingan dengan model tertentu ketika menggunakan TensorRT dan OpenVINO serta YOLOv8n dan YOLOv11n, sehingga total dilakukan lima kali pengujian dengan rincian lima indikator perbandingan. Dalam uji *levene* untuk penelitian ini, ditetapkan nilai *threshold alpha* adalah 0.05 dengan hipotesis nol adalah tidak ada perbedaan rata-rata signifikan pada data masing-masing atau homogen.

Tabel 4.16 Hasil Uji *Levene Two-Way* ANOVA

Category	***p-value***	**Status**	***Levene***
Inference speed	0.03210	Tolak H0	False
Memory consumption	0.00000	Tolak H0	False
Model size	0.00323	Tolak H0	False
Load time	0.03307	Tolak H0	False
mAP50	0.00842	Tolak H0	False

Pada tabel 4.16, disajikan data yang merupakan hasil dari uji *levene* untuk *Two-Way* ANOVA denan masing-masing merupakan pengujian terhadap variable indikator perbandingan dengan empat set data, TensorRT dengan YOLOv8n, OpenvVINO dengan YOLOv8n, TensorRT dengan YOLOv11n, OpenvVINO dengan YOLOv11n. Pada indikator *inference speed*, hasil uji *levene* menunjukkan bahwa *p-value* kurang dari nilai *threshold alpha*, yang berarti terdapat perbedaan signifikan pada keempat data yang diberikan atau data tidak bersifat homogen. Berikutnya pada indikator *memory consumption* juga didapatkan hasil *p-value* kurang dari *threshold alpha*, yang berarti pada data terdapat perbedaan yang signifikan atau data tidak bersifat homogen. Berikutnya melalui indikator *model size* kembali didapatkan nilai *p-value* yang kurang dari *threshold alpha* yang menunjukkan bahwa terdapat perbedaan signifikan pada keempat data atau data

tidak bersifat homogen. Pada indikator *load time* juga menghasilkan nilai *p-value* yang kurang dari *threshold alpha* dari uji *levene*, yang berarti ditemukan adanya perbedaan signifikan pada keempat data atau data tidak bersifat homogen. Dari uji *levene* untuk indikator *accuracy* melalui mAP50 kembali memberikan *p-value* yang kurang dari *threshold alpha* yang telah ditentukan, yang berarti ditemukan perbedaan signifikan pada keempat data atau data tidak bersifat homogen.

4.5.3.2 Hasil Uji *Two-Way* ANOVA

Melalui uji *levene* untuk *Two-Way* ANOVA, ditemukan bahwa pada keempat data ditemukan perbedaan yang signifikan atau seluruh data tidak bersifat homogen pada setiap pengujian indikator perbandingannya, sehingga pengujian *Two-Way* ANOVA dilakukan metode non parametrik. Pada penelitian ini, *Two-Way* ANOVA dengan metode non parametrik dilakukan dengan bantuan program python dengan *library pingouin* dan fungsi *anova()*. Fungsi *anova()* milik *pingouin* menerima empat buah parameter, yaitu data, *dependent variable* atau *dv*, faktor uji atau *between*, dan *detailed* untuk merincikan hasil pengujian dalam bentuk tabel. Parameter data untuk penelitian ini adalah data dari keempat kombinasi data bersama dengan nilai dari indikator perbandingannya, keempat kombinasi data adalah TensorRT dengan YOLOv8n, OpenVINO dengan YOLOv8n, TensorRT dengan YOLOv11n, OpenvVINO dengan YOLOv11n. Indikator perbandingannya akan menjadi *dependent variable*, kemudian membuat model dan *inference framework* sebagai parameter *between*. Pada parameter *detailed* akan diberi nilai *True* agar keluaran dari uji *Two-Way* ANOVA tersaji dalam bentuk tabel terperinci.

Pada tabel 4.17 disajikan hasil dari *Two-Way* ANOVA yang membandingkan performa dari *inference framework* TensorRT milik NVIDIA dengan OpenVINO milik Intel dengan setiap pemodelan YOLOv8n dan YOLOv11n. *Two-Way* ANOVA pada penelitian kali ini dilakukan terhadap setiap indikator perbandingan atau *category* yang telah disebutkan sebelumnya, sehingga total hasil dari *Two-Way* ANOVA ini adalah lima baris data hasil pengujian terhadap lima indikator perbandingan. Pada *Two-Way* ANOVA untuk penelitian ini, *threshold alpha* ditetapkan menjadi 0.05 dengan hipotesis nol bahwa tidak terdapat interaksi pada rata-rata dengan *inference framework* dan model tertentu terhadap setiap indikator perbandingan.

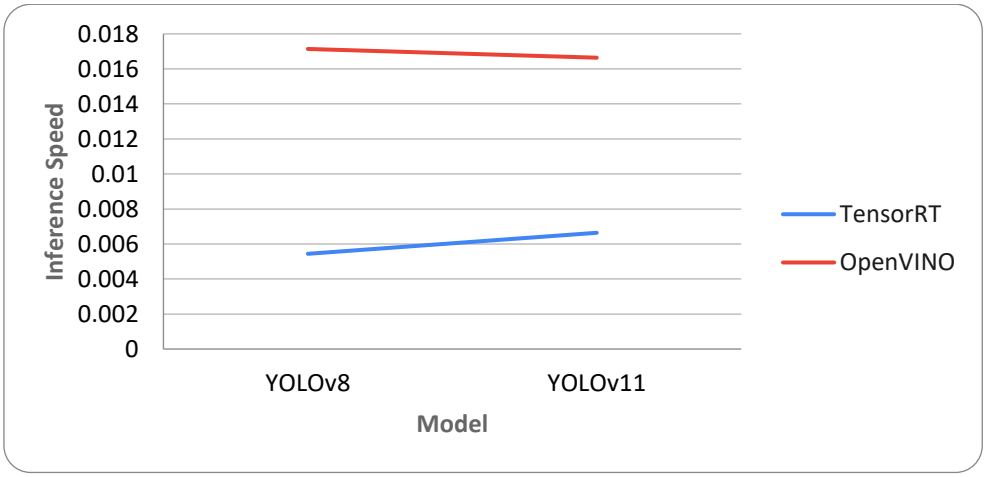
Tabel 4.17 Hasil Uji *Two-Way* ANOVA

Category	***p-value***	**Status**	**Interaksi**
Inference speed	0.00000	Tolak H0	Ada
Memory consumption	0.17109	Gagal Tolak H0	Tidak Ada
Model size	0.00090	Tolak H0	Ada
Load time	0.37492	Gagal Tolak H0	Tidak Ada
mAP50	0.20110	Gagal Tolak H0	Tidak Ada

Pada tabel 4.17 dapat dilihat bahwa indikator *inference speed* memiliki *p-value* yang kurang dari *threshold alpha* melalui uji *Two-Way* ANOVA untuk data kombinasi antara TensorRT dengan YOLOv8n, OpenvVINO dengan YOLOv8n, TensorRT dengan YOLOv11n, OpenvVINO dengan YOLOv11n, yang berarti terdapat interaksi yang signifikan antara *inference framework* dengan model tertentu terhadap rata-rata dari *inference speed*. Dengan kata lain, *inference speed* pada *inference framework* tertentu ternyata dipengaruhi oleh model tertentu. Pada indikator *memory consumption* ditemukan bahwa *p-value* bernilai 0.171 yang berarti lebih dari *threshold alpha* dan menunjukkan bahwa tidak ada interaksi signifikan pada *memory consumption* antara *inference framework* dengan model. Tidak adanya interaksi pada variabel *memory consumption* memiliki arti bahwa *inference framework* pada setiap model menunjukkan hasil yang konsisten pada indikator *memory consumption*. Melalui uji *Two-Way* ANOVA untuk *model size*, didapatkan hasil *p-value* kurang dari *threshold alpha*, yang berarti bahwa terdapat interaksi signifikan antara *inference framework* dengan model tertentu atau dapat dipahami bahwa *model size* berpengaruh terhadap sebuah *inference framework* yang memiliki model spesifik. Pada indikator *load time* didapatkan hasil *p-value* yang lebih dari *threshold alpha* yang dapat diartikan bahwa tidak terdapat interaksi signifikan dari *dependent variable load time* terhadap kombinasi data yang juga menunjukkan bahwa *load time* untuk setiap kombinasi *inference framework* dan model konsisten. Pada uji *Two-Way* ANOVA untuk *accuracy* dengan mAP50, didapatkan *p-value* yang juga lebih dari *threshold alpha* yang telah ditentukan

sebelumnya dan mengindikasikan tidak ada interaksi signifikan. Dalam hal ini, ditemukan bahwa performa *accuracy* cukup konsisten ketika menggunakan tiap *inference framework* dengan tiap model.

Setelah mendapatkan hasil *Two-Way* ANOVA, untuk mengetahui keberadaan dan kejelasan mengenai interaksi antara *inference framework* dengan model untuk setiap indikator perbandingan, perlu dilakukan kajian lebih mendalam mengenai interaksi yang terdapat pada beberapa indikator untuk mengetahui performa *inference framework* secara signifikan. Pada indikator *inference speed* dan *model size*, ditemukan interaksi antara *inference framework* dengan model, sehingga perlu dikaji lebih lanjut mengenai kedua indikator perbandingan ini untuk mengetahui performa kedua *inference framework*.

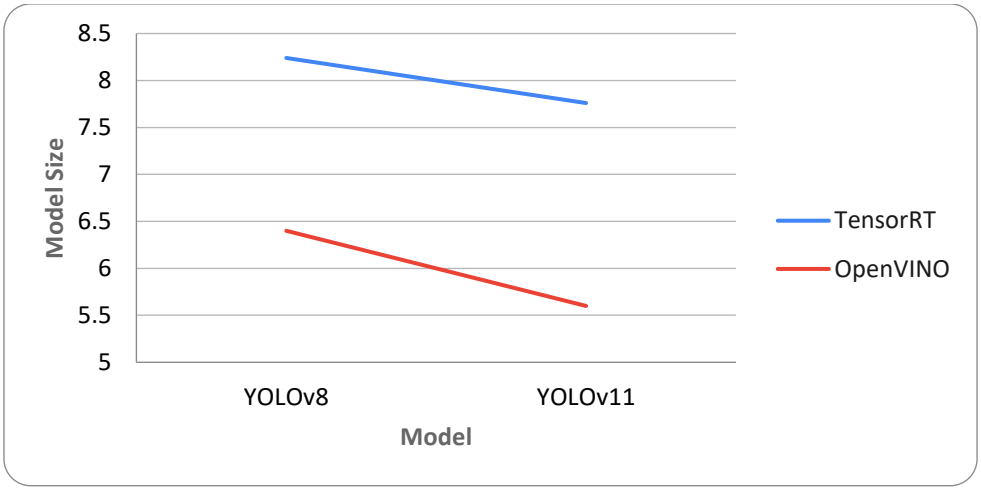


Gambar 4.3 Perbandingan *Inference Speed*

Gambar 4.7 menunjukkan rata-rata *inference speed* untuk TensorRT dan OpenVINO pada model YOLOv8n dengan YOLOv11n. Garis berwarna biru menunjukkan rata-rata dari *inference speed* ketika menggunakan YOLOv8n dan YOLOv11n, sedangkan garis berwarna merah menunjukkan *inference speed* ketika menggunakan *inference framework* OpenVINO. TensorRT menunjukkan *inference speed* yang bertambah dari 0.0054 ketika menggunakan YOLOv8n menjadi 0.0066 ketika menggunakan YOLOv11n. Hal ini mengindikasikan bahwa dalam menggunakan model yang lebih sederhana, TensorRT lebih cepat dalam melakukan *inference* dan sedikit lebih lambat ketika menggunakan model yang lebih kompleks seperti YOLOv11n. Berikutnya untuk *inference framework* OpenVINO, berlaku kebalikan dari TensorRT. *Inference speed* dari OpenVINO ketika menggunakan

model YOLOv11n mengalami penurunan dari YOLOv8n yang awalnya 0.017 menjadi 0.016 pada YOLOv11n. Penurunan *inference speed* pada YOLOv11n menunjukkan bahwa OpenVINO mampu melakukan *inference* lebih baik atau lebih cepat ketika menggunakan model yang lebih kompleks. Namun, selisih *inference speed* OpenVINO YOLOv8n dan YOLOv11n sangatlah sedikit dengan selisih hanya sebesar 0.001 yang juga menunjukkan bahwa OpenVINO lebih konsisten daripada TensorRT untuk model yang lebih kompleks, walaupun secara keseluruhan *inference framework* TensorRT masih lebih cepat sekitar tiga kali lipat daripada OpenVINO. Dengan ini, *inference framework* TensorRT tetap dapat menjadi pertimbangan untuk kebutuhan kecepatan *inference* seperti pada kasus *real-time processing*, sedangkan OpenVINO dapat digunakan untuk kebutuhan penggunaan model yang lebih kompleks tanpa mempertimbangkan kebutuhan *real-time processing*.

Melalui penelitian ini, tedapat temuan mengenai *inference speed* dari model YOLOv8n dan model YOLOv11n saat menggunakan *inference framework* TensorRT dan OpenVINO yang menunjukkan kestabilan dari kemampuan TensorRT untuk melakukan *inference* dengan baik dan kecepatan yang tinggi



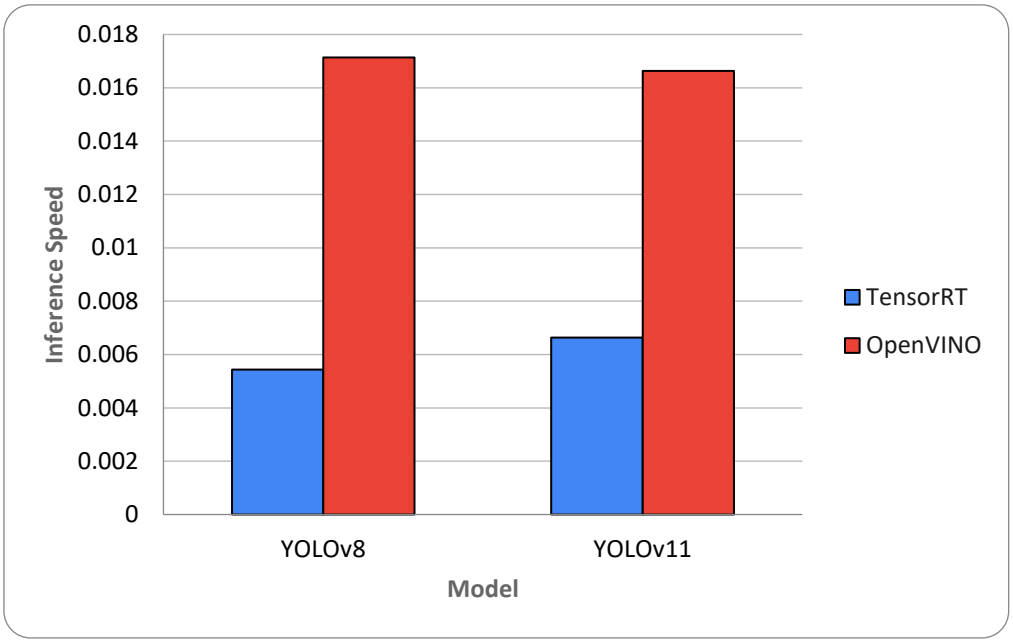
Gambar 4.4 Perbandingan *Model Size*

Gambar 4.4 menunjukkan rata-rata dari *model size* untuk TensorRT dan OpenVINO ketika menggunakan model YOLOv8n dan YOLOv11n. Garis berwarna biru menunjukkan rata-rata dari *model size* ketika menggunakan YOLOv8n dan YOLOv11n, sedangkan garis berwarna merah menunjukkan *model*

size ketika menggunakan *inference framework* OpenVINO. Melalui gambar 4.8, dapat dilihat bahwa kedua *inference framework* mengalami penurunan ukuran model ketika menggunakan model YOLOv11n yang menujukkan bahwa dengan kompleksitas model milik YOLOv11n, ternyata tidak membuat model YOLOv11n berukuran lebih besar daripada model milik YOLOv8n setelah dilakukan *export* menjadi *inference framework*. Hal ini mengindikasikan bahwa YOLOv11n lebih efisien daripada YOLOv8n. Kajian lebih mendalam mengenai perbandingan kedua *inference framework* ini dapat dilihat dari selisih ukuran dari model milik YOLOv8n dan YOLOv11n ketika menggunakan TensorRT dan OpenVINO. Selisih *model size* dari ketika menggunakan TensorRT dari YOLOv8n dan YOLOv11n adalah sebesar 0.48 MB, sedangkan pada penggunaan *inference framework* OpenVINO selisihnya adalah 0.8 MB. Hal ini menunjukkan bahwa OpenVINO mampu melakukan konversi model atau *export* model lebih baik daripada TensorRT pada model yang kompleks, karena mampu membuat ukuran model YOLOv11n menjadi lebih kecil daripada YOLOv8n dengan penurunan ukuran sebesar 0.8 MB. Secara keseluruhan, *inference framework* OpenVINO mengindikasikan adanya keunggulan pada *model size*, sehingga sesuai untuk kebutuhan perangkat yang terbatas pada *storage* serta OpenVINO juga unggul dalam melakukan konversi atau *export* model yang kompleks dengan lebih maksimal daripada TensorRT.

4.5.4 Pembahasan Hasil Pengujian

Melalui pengujian statistik *T-Test* yang diikuti dengan *Two-Way* ANOVA dapat dilihat bahwa pada *inference framework* TensorRT memberikan performa yang lebih optimal pada indikator *inference speed* dibandingkan dengan OpenVINO. Melalui kajian lebih mendalam berdasarkan data pada tabel 4.8 juga ditemukan bahwa TensorRT lebih unggul 0.0012 detik secara *inference speed* pada model YOLOv8n dibandingkan YOLOv11n dengan *inference framework* TensorRT, sedangkan OpenVINO unggul ketika menggunakan YOLOv11n sebesar 0.0005 detik dibandingkan YOLOv8n dengan OpenVINO.

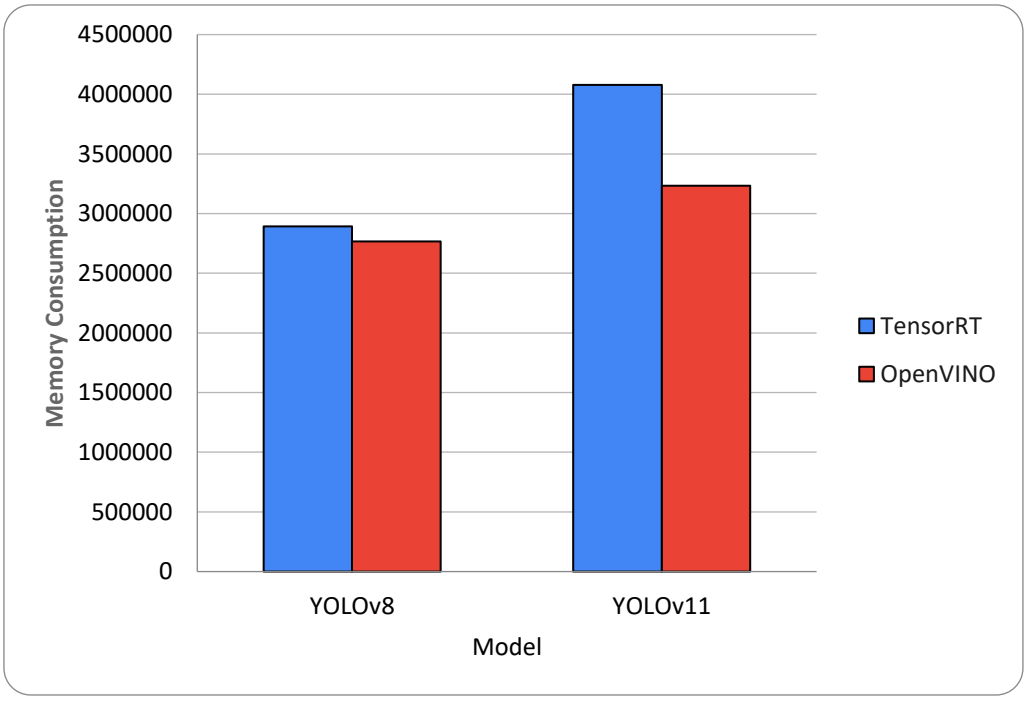


Gambar 4.5 Grafik Perbandingan Rata-rata *Inference Speed*

Melalui gambar 4.5 dapat dilihat bahwa TensorRT dapat melakukan *inference* dengan cepat untuk YOLOv8n maupun untukYOLOv11n dibandingkan dengan OpenVINO dengan selisih sekitar tiga kali lipat lebih cepat.. Disisi lain, dapat dipahami juga bahwa OpenVINO memiliki kemampuan yang lebih stabil jika dibandingkan dengan TensorRT dalam melakukan *optimization* terhadap model yang sederhana maupun kompleks, dilihat dari selisih pada *inference speed* OpenVINO antara kedua model yang lebih kecil daripada selisih *inference speed* pada TensorRT. Pada indikator *inferefence speed*, dapat disimpulkan bahwa TensorRT lebih unggul dan sesuai untuk kebutuhan *real-time processing* terlebih jika menggunakan model yang sederhana seperti YOLOv8n, tetapi jika ingin menggunakan model yang lebih kompleks tanpa mempertimbangkan *inference speed*, OpenVINO dapat menjadi pilihan yang sesuai. Hal ini sesuai dengan temuan dari penelitian Chaturvedi (2022) mengenai TensorRT yang mampu melakukan *inference* tiga kali lebih cepat.

Hasil pengujian statistik *T-Test* yang diikuti dengan *Two-Way* ANOVA menunjukkan bahwa pada *inference framework* TensorRT dan OpenVINO ditemukan suatu persamaan pada indikator *memory consumption*, dimana saat menggunakan model yang lebih kompleks seperti YOLOv11n, kebutuhan akan

memory meningkat cukup signifikan hingga setidaknya 500000 MB pada OpenVINO dan lebih dari 1000000 MB pada TensorRT. Dari data *memory consumption* pada tabel 4.9 saat menggunakan TensorRT dan OpenVINO, dapat dipahami bahwa meskipun kedua *inference framework* mengindikasikan adanya peningkatan *memory consumption* pada model yang lebih kompleks seperti YOLOv11n dibandingkan model lebih sederhana seperti YOLOv8n, tetapi kenaikan *memory consumption* ketika menggunakan OpenVINO hanya sekitar 16 % sedangkan saat menggunakan TensorRT kenaikannya adalah sekitar 41%.

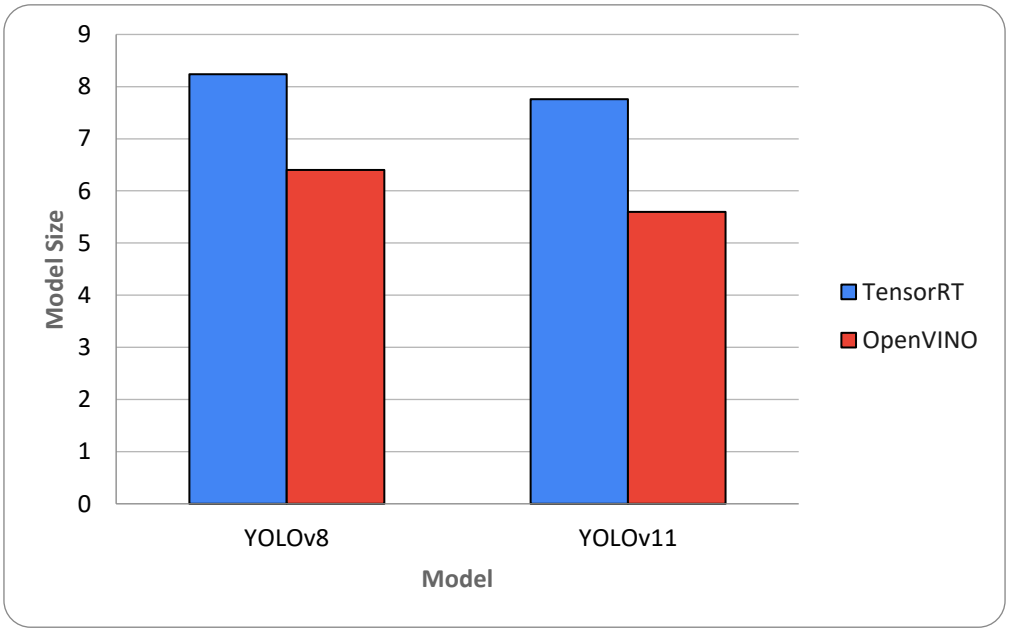


Gambar 4.6 Grafik Perbandingan Rata-rata *Memory Consumption*

Melalui gambar 4.6 dapat dilihat bahwa TensorRT dan OpenVINO sama-sama mengalami peningkatan *memory consumption* saat menggunakan model yang lebih kompleks. Namun, melalui indikator *memory consumption*, dapat disimpulkan bahwa OpenVINO sedikit lebih unggul baik untuk model sederhana maupun kompleks seperti YOLOv11n dan lebih sesuai untuk kebutuhan *processing* yang memiliki keterbatasan pada perangkat keras seperti RAM dikarenakan kemampuan OpenVINO dalam mengoptimalkan model sederhana dan kompleks yang lebih stabil dibandingkan dengan TensorRT dengan peningkatan sebesar 16%

untuk OpenVINO dan 40% untuk TensorRT Ketika menggunakan model yang lebih kompleks.

Hasil dari pengujian statistik *T-Test* yang diikuti dengan *Two-Way* ANOVA menunjukkan bahwa TensorRT dan OpenVINO mampu melakukan *optimization* pada model dengan baik, terlebih pada model YOLOv11n yang terbilang lebih kompleks dibandingkan dengan YOLOv8n. Hal ini menunjukkan bahwa YOLOv11n dengan kompleksitasnya ternyata tidak mempengaruhi ukuran model. Dari data *model size* pada tabel 4.10 saat menggunakan TensorRT dan OpenVINO, dapat dipahami bahwa OpenVINO mampu melakukan konversi atau *export* model *deep learning* dengan lebih baik saat menggunakan YOLOv8n maupun YOLOv11n karena hasil konversi modelnya lebih ringan daripada saat menggunakan *inference framework* TensorRT.

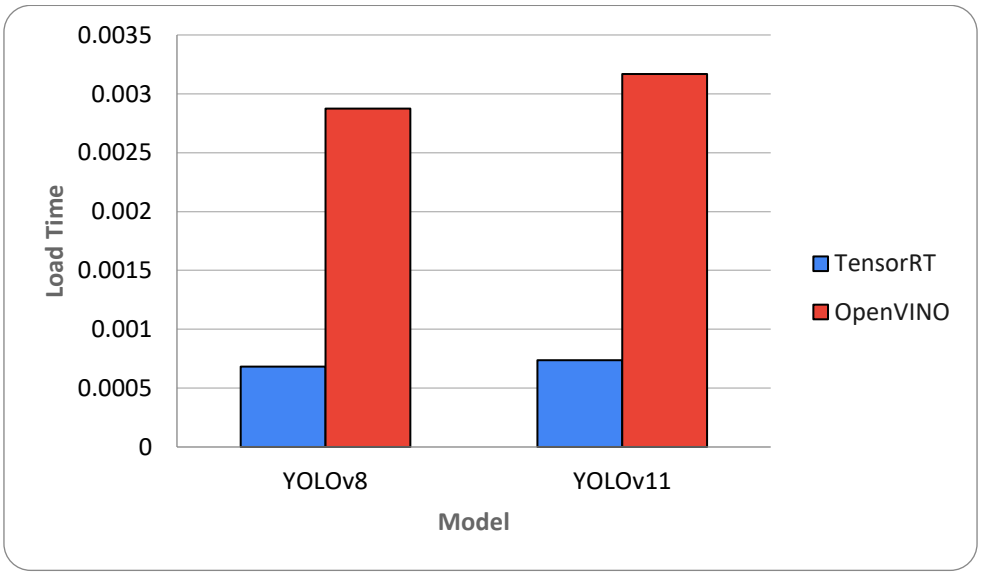


Gambar 4.7 Grafik Perbandingan Rata-rata *Model Size*

Gambar 4.7 menujukkan perbandingan yang signifikan antara *inference framework* TensorRT dan OpenVINO pada indikator *model size*, dimana TensorRT lebih konsisten dalam melakukan konversi model menjadi lebih ringan dibandingkan dengan OpenVINO karena memiliki selisih hanya sebesar 0.48 MB ketika menggunakan model sederhana maupun kompleks. Dari data juga dapat disimpulkan bahwa OpenVINO sangat sesuai untuk kebutuhan yang mempertimbangkan perangkat keras khususnya *storage*. Model YOLOv11n

dengan kompleksitasnya ternyata juga tidak menujukkan peningkatan pada *model size*, tetapi justru menawarkan *model size* yang lebih ringan. Pada penelitian ini, mengenai *memory consumption* dan *model size* terbukti selaras dengan penelitian terdahulu oleh Li (2022) yang juga menyatakan bahwa secara keseluruhan, OpenVINO mampu memberikan performa yang lebih baik dan *power efficiency* pada banyak skenario.

Melalui hasil pengujian statistik dengan *T-Test* dan *Two-Way* ANOVA ditemukan bahwa OpenVINO membutuhkan waktu yang lebih lama dalam melakukan *load model* dibandingkan dengan TensorRT. Namun, pada kedua *inference framework* ditemukan kesamaan bahwa model yang lebih kompleks seperti YOLOv11n membutuhkan waktu lebih lama untuk *load model* daripada model lebih sederhana seperti YOLOv8n. Dari data *load time* pada tabel 4.11 saat menggunakan TensorRT dan OpenVINO, dapat dipahami bahwa TensorRT mampu melakukan membuat model *deep learning* lebih cepat untuk dimuat sekitar 300%, baik saat menggunakan model YOLOv8n maupun YOLOv11n dibandingkan dengan menggunakan *inference framework* OpenVINO.

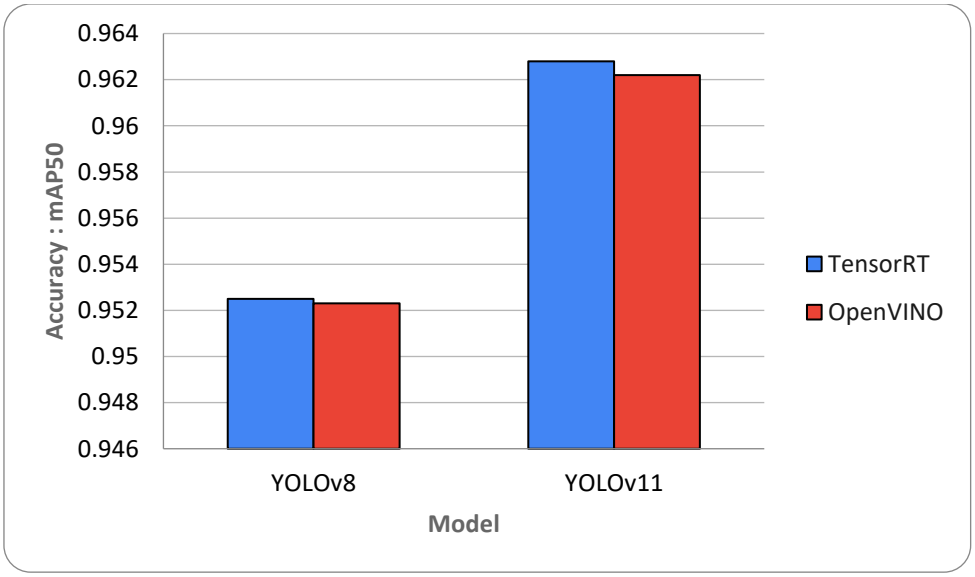


Gambar 4.8 Grafik Perbandingan Rata-rata *Model Load Time*

Dari gambar 4.8 dapat dilihat dengan lebih jelas mengenai performa TensorRT dalam memuat model. *Load time* untuk model ketika menggunakan *inference framework* TensorRT sangat baik untuk model sederhana seperti YOLOv8n maupun kompleks seperti YOLOv11n. TensorRT juga stabil dalam

melakukan *load model*, yang dapat dilihat dari selisih *load time* untuk model sederhana maupun kompleks sebesar 0.00015 detik, sedangkan 0.003 detik untuk OpenVINO. Hal ini dapat diartikan bahwa TensorRT unggul ketika membutuhkan kecepatan dalam proses *deployment*. Hasil dari penelitian yang dilakukan oleh Zhou dan Yang (2022) kembali selaras dengan temuan pada penelitian ini bahwa TensorRT mampu memberikan *inference speed* dan *load time model* yang sangat baik sehingga sesuai dengan kebutuhan performa atau kecepatan *inference* yang tinggi.

Dari pengujian statistik yang melalui *T-Test* dan *Two-Way* ANOVA, ditemukan bahwa pada *inference framwork* TensorRT dan OpenVINO terlihat hal yang sama pada *accuracy* yang meningkat pada penggunaannya untuk model yang lebih kompleks seperti YOLOv11n. Dari data *accuracy* dengan mAP50 saat menggunakan TensorRT dan OpenVINO, dapat dipahami bahwa YOLOv11n memberikan *accuracy* yang lebih tinggi dibandingkan dengan YOLOv8n. Dari sini dapat diartikan bahwa dengan kompleksitas yang dimiliki YOLOv11n, ternyata mampu membawa dampak peningkatan dalam segi *accuracy* melalui mAP50 jika dibandingkan dengan YOLOv8n.



Gambar 4.9 Grafik Perbandingan Rata-rata *Accuracy*

Gambar 4.9 menunjukkan bahwa dengan jelas bahwa *accuracy* TensorRT pada YOLOv8n sedikit lebih tinggi dibandingkan menggunakan OpenVINO. Saat

menggunakan YOLOv11n, terlihat hal yang sama, dimana TensorRT memberikan *accuracy* yang lebih tinggi. Namun, OpenVINO mampu melakukan konversi yang cukup stabil pada model yang lebih sederhana seperti YOLOv8n dan model yang lebih kompleks seperti YOLOv11n dengan selisih sebesar 0.99% jika dibandingkan selisih pada TensorRT pada model yang lebih kompleks sebesar 1.03%. Dari gambar dapat disimpulkan bahwa *inference framework* TensorRT mampu memberikan performa *accuracy* yang sedikit lebih tinggi jika dibandingkan dengan OpenVINO pada model sederhana seperti YOLOv8n maupun model yang lebih kompleks seperti YOLOv11n. Peningkatan *accuracy* yang ditemukan pada penelitian ini dari perbandingan YOLOv8n dan YOLOv11n juga selaras dengan temuan pada penelitian-penelitian terdahulu. YOLOv11 memberikan performa optimal yang seimbang antara *accuracy*, *efficiency*, dan secara *model size* (Jegham, *et al*, 2024) dan penelitian oleh Khanam dan Hussein (2024) yang turut menyatakan bahwa YOLOv11 merupakan perkembangan dari *feature extraction*, *optimized performance*, dan kemampuannya pada berbagai *task* dalam *computer vision*.

BAB V
Simpulan dan Saran

5.1 Kesimpulan

Pada penelitian tugas akhir dengan judul "Analisis Kinerja TensorRT dan OpenVINO sebagai *Inference Framework* untuk Pemodelan *Deep Learning*" telah berjalan dengan baik. Melalui penelitian yang telah dilakukan, dapat diperoleh beberapa kesimpulan sebagai berikut.

1. *Inference framework* TensorRT lebih unggul dibandingkan OpenVINO jika dilihat performa *inference speed* dan *load time*. Sehingga TensorRT sangat sesuai untuk kebutuhan aplikasi model *deep learning* yang sangat memperhatikan kecepatan atau *real-time processing* tanpa memperhatikan batasan perangkat seperti *storage* dan *memory*.
2. *Inference framework* OpenVINO lebih unggul dibandingkan TensorRT jika melihat performa *memory consumption* dan *model size*. OpenVINO ditemukan membutuhkan RAM dan *storage* yang lebih sedikit dikarenakan kemampuan OpenVINO untuk melakukan *export* model menjadi lebih sederhana. OpenVINO sesuai untuk kebutuhan aplikasi *deep learning* yang mempertimbangkan keterbatasan pada perangkat keras seperti *storage* dan *memory* dengan mengesampingkan *inference speed*.
3. Untuk indikator perbandingan *accuracy* dengan menggunakan mAP50, tidak dapat dilihat perbedaan yang signifikan. Namun, dengan kompleksitas modelnya, YOLOv11n selalu mampu memberikan performa yang 1% lebih baik dan OpenVINO yang memiliki *accuracy* lebih rendah dari TensorRT sebesar 0.04%.
4. Melalui pengujian *Two-Way* ANOVA, ditemukan bahwa terdapat interaksi yang cukup signifikan pada indikator *inference speed* dan *model size* antara model dengan *inference framework*. TensorRT memang lebih baik dari indikator *inference speed*, tetapi ketika digunakan pada model yang lebih kompleks, TensorRT akan sedikit mengalami kemunduran dalam *inference speed*, sedangkan OpenVINO akan mengalami kemajuan

atau lebih cepat dalam *inference speed* saat menggunakan model yang lebih kompleks. OpenVINO memiliki kemampuan konversi model yang jauh lebih baik dibandingkan TensorRT, tetapi secara khsus, OpenVINO yang digunakan pada model yang kompleks mampu menghasilkan model yang lebih ringan.

5. TensorRT secara keseluruhan lebih sesuai untuk kasus *counting box* secara *real-time* pada *conveyor belt* yang sedang diterapkan oleh PT. XYZ dikarenakan keunggulannya dalam melakukan *inference* secara *real-time* yang berkaitan dengan kecepatan dari *conveyor belt* keluaran produksi. Namun, apabila terdapat kendala dalam *cost* untuk pengadaan *computing device* dan kebutuhan *real-time counting* dapat dikesampingkan, maka OpenVINO dapat menjadi opsi yang bisa dipertimbangkan oleh PT. XYZ.

5.2 Saran

Melalui kesimpulan dari penelitian yang telah disebutkan sebelumnya, terdapat beberapa saran yang dapat dilakukan untuk memperbaiki dan mengembangkan penelitian ini.

1. PT. XYZ disarankan menggunakan *inference framework* milik TensorRT dengan *computing device* GPU NVIDIA dan model YOLOv8n dikarenakan terdapat kebutuhan *counting box* secara *real-time* pada *conveyor belt* keluaran produksi, karena TensorRT mampu melakukan *inference* dengan rata-rata kecepatan sebesar 0.006 detik dan memiliki *memory consumption* yang lebih rendah daripada penggunaan YOLOv11n.
2. Dalam penelitian ini, pemodelan *deep learning* yang digunakan adalah YOLOv8n dengan YOLOv11n yang sama-sama dikembangkan oleh tim *ultralytics*. Akan lebih baik jika dalam penelitian berikutnya, pemodelan *deep learning* yang digunakan memiliki arsitektur yang berbeda signifikan, agar hasil perbedaan terlihat lebih jelas.
3. Pada penelitian ini, validasi terhadap datanya dilakukan melalui metode *val()* yang merupakan fungsi dari *ultralytics*, disarankan validasi data untuk penelitian berikutnya adalah menggunakan data nyata, atau *real-world use case*.

Daftar Pustaka

Ahn, H., Chen, T., Alnaasan, N., Shafi, A., Abduljabbar, M., Subramoni, H., Panda, D. K. (2023). Performance Characterization of using Quantization for DNN Inference on Edge Devices: Extended Version. *ArXiv*, 1-5.

Al Ghadani, A. K. A., Mateen, W., Ramaswamy, R. G. (2020). Tensor-Based CUDA Optimization for ANN Inferencing Using Parallel Acceleration on Embedded GPU. *Springer Nature Switzerland AG*, 2-9.

Arinez, J. F., Chang, Q., Gao, R. X., Xu, C., Zhang, J. (2020). Artificial Intelligence in Advanced Manufacturing: Current Status and Future Outlook. *J. Manuf. Sci. Eng*, 2-3.

Assunção, E., Gaspar, P. D., Mesquita, R., Simões, M. P., Alibabaei, K., Veiros, A., Proença, H. (2022). Real-Time Weed Control Application Using a Jetson Nano Edge Device and a Spray Mechanism. *Remote Sensing*, 8-17.

Ats-Tsauri, M. I., Setiawan, R., dan Wiyatno, T. (2021). Tinjauan Literatur Sistematis Implementasi Total Quality Management pada Industri Manufaktur - Tren Terkini dan Arahan Masa Depan. *Jurnal Teknik Industri*, 65.

Awad, A., Hegazy, M., Aly, S. A. (2024). Early Diagnoses of Acute Lymphoblastic Leukemia Using YOLOv8 and YOLOv11 Deep Learning Models. *ArXiv*, 4.

Biswas, R., Basu, A., Nandy, A., Deb, A., Chowdhury, R., Chanda, D. (2020). Identification of Pathological Disease in Plants using Deep Neural Networks - Powered by Intel® Distribution of OpenVINO™ Toolkit. *Indo-Taiwan 2nd International Conference on Computing, Analytics and Networks (Indo-Taiwan ICAN)* (pp. 45-48). Punjab, India: IEEE.

Chaturvedi, P., Khan, A., Tian, M., Huerta, E. A., Zheng, H. (2022). Inference-optimized AI and High Performance Computing for Gravitational Wave Detection at Scale. *ArXiv*, 11.

Du, Y., Chen, Z., J, C., Yin, X., Zheng, T., Li, C., Du, Y., Jiang, Y. (2022). SVTR: Scene Text Recognition with a Single Visual Model. *arXiv*, 6.

Haenlein, M., Kaplan, A. (2019). A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence. *California Management Review*, 1-10.

Hao, Z. (2018). Deep Learning Review and Discussion of Its Future Development. *MATEC Web of Conferences*, 2.

Hinton, G., Vinyals, O., Dean, J. (2015). Distilling the Knowledge in a Neural Network. *ArXiv*, 2.

Huang, H. (2023, April 18). *open-mmlab*. Retrieved from GitHub: https://github.com/open-mmlab/mmyolo/tree/main/configs/yolov8

Janiesch, C., Zschech, P., Heinrich, K. (2021). Machine learning and Deep Learning. *Springer*, 5.

Jegham, N., Koh, C. Y., Abdelatti, M., Hendawi, A. (2024). Evaluating the Evolution of YOLO (You Only Look Once) Models: A Comprehensive Benchmark Study of YOLO11 and Its Predecessors. *ArXiv*, 18.

Khanam, R., Hussain, M. (2024). YOLOv11: An Overview of the Key Architectural Enhancements. *ArXiv*, 8.

Kim, J., Chang, S., Kwak, N. (2021). PQK: Model Compression via Pruning, Quantization, and Knowledge Distillation. *ArXiv*, 1-2.

Li, H. (2022). Acceleration of Deep Learning Applications using Intel Distribution of OpenVINO Toolkit. *University of Illinois Urbana-Champaign*, 35.

Liu, X., Li, Y. (2024). A Multiscale Grouped Convolution and Lightweight Adaptive Downsampling-Based Detection of Protective Equipment for Power Workers. *Electronics*, 17.

Lubis, M. S. (2021). Implementasi Artificial Intelligence pada System Manufaktur Terpadu. *Jurnal Teknologi Industri, Fakultas Teknik - Universitas Tarumanegara*, 7.

Luo, B., Kou, Z., Han, C., Wu, J. (2023). A “Hardware-Friendly” Foreign Object Identification Method for Belt Conveyors Based on Improved YOLOv8. *Appl. Sci*, 17.

Mani, C., Paul, T. S., Archambault, P. M., Marois, A. (2024). Machine Learning Workflow for Edge Computed Arrhytmia Detection in Exploration Class Missions. *Microgravity*, 7.

O'Neill, J., Ver Steeg, G., Galstyan, A. (2020). Compressing Deep Neural Networks via Layer Fusion. *ArXiv*, 7.

Pochelu, P. (2022). Deep Learning Inference Frameworks Benchmark. *ArXiv*, 1.

Prasetyo H. dan Sutopo, W. (2018). Industri 4.0: Telaah Klasifikasi Aspek dan Arah Perkembangan Riset. *J@ti Undip: Jurnal Teknik Industri*, 24.

Ramadhani, A., Polem, A. M., Zahra, S. S. (2022). Konsep Dasar Uji T Dalam Statistika Pendidikan. *Al Ittihadu*, 1.

Rao, S. N. (2024, October 23). *YOLOv11 Architecture Explained: Next-Level Object Detection with Enhanced Speed and Accuracy*. Retrieved from Medium: https://medium.com/@nikhil-rao-20/yolov11-explained-next-level-object-detection-with-enhanced-speed-and-accuracy-2dbe2d376f71

Redmon, J. Divvala, S., Girshick, R., Farhadi, A. (2016). You Only Look Once : Unified, Real-Time Object Detection. *Computer Vision Pattern and Recognition*, 780-781.

Reis, D., Hong, J., Kupec, J., Daoudi, A. (2024). Real-Time Flying Object Detection with YOLOv8. *ArXiv*, 8-9.

Swathi, M., Dhayalakrishnan, R. (2024). Bots and Books: How Artificial Intelligence is Shaping Contemporary Literature. *Contemporaneity of English Language and Literature in the Robotized Millennium*, 4.

Szeliski, R. (2011). Recognition. In: Computer Vision. Texts in Computer Science. *Springer*, 575.

Terven, J., Cordova-Esparza, D., Romero-Gonzales, J. (2023). A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. *Machine Learning & Knowledge Extraction*, 17012-1703.

Zhongyi, L., Yih, M., Ota, J. M., Owens, J. D., Muyan-Ozcelik, P. (2019). Benchmarking Deep Learning Frameworks and Investigating FPGA Deployment for Traffic Sign Classification and Detection. *IEEE*, 11.

Zhou, Y., Yang, K. (2022). Exploring TensorRT to Improve Real-Time Inference for Deep Learning. *IEEE*, 8.