TIN: Terapan Informatika Nusantara
Vol 99, No 99, Month Year, page 999-999
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/ tin.v99i99.9999

Pengembangan Fitur Text Detection Dengan Algoritma YOLO Pada Aplikasi OCR Di PT XYZ

Yosef Manganju Manalu[1], Noper Ardi[2],
[1] Teknik Informatika, Teknik Informatika, Politeknik Negeri Batam, Batam, Indonesia
[2] Teknik Informatika, Teknologi Rekayasa Perangkat Lunak, Politeknik Negeri Batam, Batam, Indonesia
Email: [1]yosefmanganju@gmail.com, [2]noperardi@polibatam.ac.id

Abstrak-*Optical Character Recognition* (OCR) merupakan teknologi yang mengubah teks dalam citra atau gambar menjadi data digital yang dapat dikenali oleh mesin. Dengan kemajuan kecerdasan buatan, algoritma YOLO (*You Only Look Once*) semakin banyak digunakan dalam sistem OCR karena kemampuannya mendeteksi objek kecil secara real-time dengan akurasi tinggi. Penelitian ini bertujuan mengembangkan model deteksi teks otomatis berbasis YOLO yang dapat diimplementasikan untuk mendukung proses *quality control* di PT XYZ, sebuah perusahaan manufaktur di bidang elektrifikasi dan otomasi industri. Model dilatih menggunakan dataset gambar produk industri dan dirancang untuk mendeteksi serta menandai posisi teks secara otomatis. Hasil implementasi menunjukkan bahwa sistem mampu membaca citra dari folder secara otomatis, mendeteksi teks dengan tingkat akurasi yang baik, dan menghasilkan keluaran berupa gambar beranotasi serta file koordinat dalam format txt. Evaluasi dilakukan menggunakan pendekatan functional testing dan menunjukkan performa sistem berjalan sesuai harapan. Namun, nilai *Box Loss* yang cukup tinggi mengindikasikan bahwa presisi posisi *bounding box* masih perlu ditingkatkan, terutama melalui penambahan jumlah dan keragaman data pelatihan. Pengembangan ini menunjukkan efektivitas YOLO dalam mendeteksi teks, serta potensial diterapkan dalam proses inspeksi otomatis di sektor industri manufaktur.

Kata Kunci: Text Detection; OCR; Quality; YOLO; Deep Learning

Abstract-Optical Character Recognition (OCR) is a technology that converts text within images into digital data that can be recognized by machines. With advancements in artificial intelligence, the YOLO (You Only Look Once) algorithm has become increasingly used in OCR systems due to its ability to detect small objects in real-time with high accuracy. This study aims to develop an automated text detection model based on YOLO, which can be implemented to support quality control processes at PT XYZ, a manufacturing company in the field of electrification and industrial automation. The model was trained using a dataset of industrial product images and designed to automatically detect and mark text positions. The implementation results show that the system can automatically read images from a folder, detect text with good accuracy, and produce outputs in the form of annotated images and coordinate files in txt format. Evaluation was conducted using a functional testing approach and demonstrated that the system performs as expected. However, a relatively high Box Loss value indicates that the precision of bounding box positions still needs improvement, particularly through the addition of more diverse training data. This development demonstrates the effectiveness of YOLO in text detection and its potential for application in automated inspection processes within the manufacturing industry.

Keywords: Text Detection; OCR; Quality; YOLO; Deep Learning

1. PENDAHULUAN

Optical Character Recognition (OCR) merupakan teknologi yang memungkinkan konversi karakter dalam bentuk citra atau gambar menjadi teks digital yang dapat dikenali dan diproses oleh mesin. Dalam beberapa tahun terakhir, OCR telah mengalami perkembangan signifikan berkat integrasinya dengan kecerdasan buatan dan *deep learning*, yang secara drastis meningkatkan akurasi serta efisiensi dalam pengenalan teks. Salah satu algoritma terkini yang digunakan dalam pengembangan sistem deteksi teks berbasis OCR adalah *You Only Look Once* (YOLO), sebuah metode deteksi objek real-time yang terkenal karena kecepatan inferensinya serta kemampuan mendeteksi objek kecil dan kompleks secara presisi.

Menurut Maleh et al. (2023)YOLO merupakan metode detektor dengan model terpadu (*unified*), yang mana dengan jaringan saraf tunggal (*single neural network*) dapat memprediksi kotak pembatas (*Bounding Box*) dan probabilitas kelas secara langsung dalam satu gambar penuh pada sekali tangkapan . Keunggulan inilah yang menjadikan YOLO sangat relevan untuk diterapkan dalam pengembangan sistem OCR yang mampu mendeteksi teks pada produk industri secara otomatis. Penerapan OCR berbasis YOLO dalam konteks industri telah terbukti efektif, misalnya pada pengenalan teks pada papan sirkuit tercetak (PCB), di mana sistem mampu memberikan anotasi visual dan koordinat lokasi teks secara akurat (Cai, 2023).

Pengembangan teknologi OCR berbasis YOLO ini dilakukan dalam konteks praktik di PT XYZ, sebuah perusahaan manufaktur yang bergerak di bidang elektrifikasi, otomasi industri, dan sistem kontrol. Didirikan pada tahun 1991, PT XYZ telah berkembang menjadi produsen berbagai produk elektrifikasi berkualitas tinggi seperti kontaktor, relay, sensor, dan perangkat elektronik lainnya. Perusahaan ini melayani pasar domestik dan juga mengekspor produknya ke berbagai negara, termasuk kawasan Eropa dan Amerika Utara.

Dalam operasionalnya, PT XYZ sangat menekankan pentingnya quality control sebagai bagian krusial untuk memastikan produk memenuhi standar internasional. Untuk mendukung proses ini, perusahaan telah mengadopsi teknologi OCR sebagai alat bantu inspeksi otomatis. Namun, sistem OCR yang digunakan saat ini masih memiliki keterbatasan, terutama pada proses penandaan lokasi teks yang masih dilakukan secara manual. Hal ini menimbulkan ketidakefisienan, terutama karena posisi teks pada produk sering berubah-ubah, sehingga memaksa *Maintenance Engineer* untuk sering keluar-masuk area produksi guna memperbarui data lokasi teks.

TIN: Terapan Informatika Nusantara
Vol 99, No 99, Month Year, page 999-999
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/ tin.v99i99.9999

Hal ini menimbulkan ketidakefisienan, terutama karena posisi teks pada produk sering berubah-ubah, sehingga memaksa Maintenance Engineer untuk sering keluar-masuk area produksi guna memperbarui data lokasi teks. Tujuan utama dari penelitian ini adalah untuk menjawab tantangan tersebut dengan mengembangkan sebuah fitur deteksi teks otomatis yang dapat menghilangkan kebutuhan intervensi manual, sehingga proses quality control menjadi lebih cepat dan efisien.

Menanggapi tantangan tersebut, departemen *Maintenance* PT XYZ mengusulkan pengembangan fitur Text Detection otomatis berbasis algoritma YOLO menggunakan bahasa pemrograman Python. Fitur ini dirancang agar mampu mendeteksi dan memberikan output berupa gambar dengan anotasi serta koordinat lokasi teks pada produk secara otomatis. Dengan sistem ini, proses pengecekan kualitas tidak hanya menjadi lebih cepat dan efisien, tetapi juga mampu mengurangi ketergantungan pada proses manual yang selama ini memakan waktu. Penggunaan seperti teknologi ini tidak hanya mempercepat proses pemeriksaan, tetapi juga meningkatkan akurasi dan konsistensi dalam mendeteksi kesalahan, sehingga mendukung komitmen perusahaan untuk selalu menghadirkan produk berkualitas tinggi bagi pasar global (Palanikumar et al., 2024).

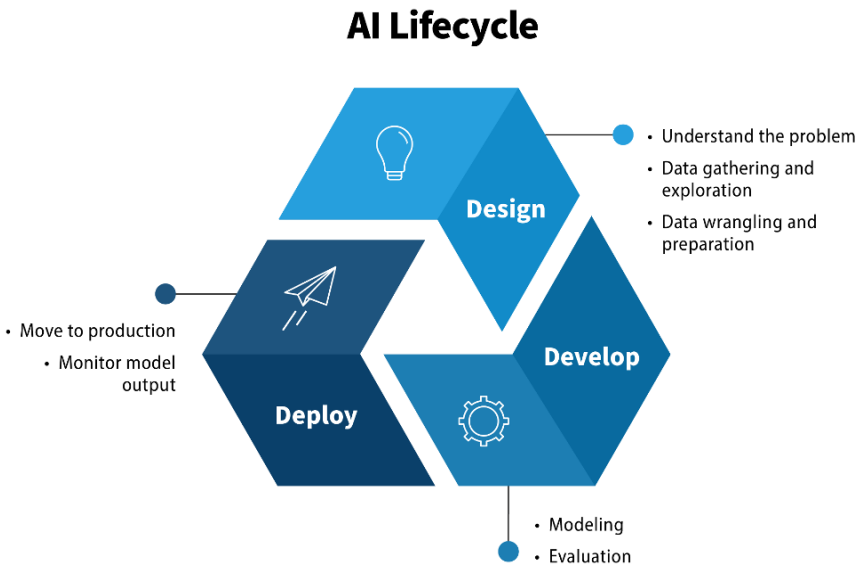
Integrasi fitur *Text Detection* berbasis YOLO ke dalam aplikasi XYZ OCR diharapkan mampu menjawab kebutuhan operasional perusahaan, sekaligus menjadi solusi yang adaptif terhadap dinamika produksi. Inovasi ini tidak hanya memperkuat kualitas produk, tetapi juga menunjukkan komitmen PT XYZ dalam menerapkan teknologi cerdas untuk mendorong transformasi digital di sektor manufaktur.

Sejumlah penelitian telah menunjukkan efektivitas algoritma YOLO dalam deteksi teks dan objek secara real-time. Ramadhan et al. (2024) mengembangkan model YOLO untuk mendeteksi plat nomor kendaraan dengan waktu inferensi cepat dan akurasi tinggi. Wang et al. (2021) mengusulkan *Rotational* YOLO (R-YOLO) untuk deteksi teks berorientasi sembarang, meningkatkan fleksibilitas OCR terhadap variasi rotasi. Haifeng & Siqi (2020) menggunakan YOLOv2 untuk membedakan latar belakang dan teks, yang meningkatkan akurasi visual. Sementara itu, Mamuriyah & Jacky, (2021), meskipun tidak menggunakan YOLO, menunjukkan adaptabilitas OCR untuk karakter multibahasa. Jonathan et al. (2023) menggabungkan YOLOv3 dan OCR untuk deteksi serta pembacaan plat nomor kendaraan. Temuan-temuan ini menguatkan bahwa integrasi YOLO dalam sistem OCR industri sangat potensial dalam mendeteksi teks yang akan meningkatkan efisiensi dan akurasi proses quality control.

2. METODE PENELITIAN

2.1 Metode Pengembangan

Metode adalah tahap-tahap ataupun aturan untuk melakukan sesuatu. Artificial Intelligence Life Cycle adalah sebuah siklus dalam proses pembuatan proyek *Artificial Intelligence* untuk dapat membantu dalam menciptakan solusi permasalahan berbasis *Artificial Intelligence*. Pada penelitian ini, siklus hidup *Artificial Intelligence* yang digunakan merupakan adaptasi dari CDAC AI *Life Cycle* yang diperkenalkan oleh De Silva & Alahakoon (2022), yang terdiri dari tiga fase utama yaitu *Design*, *Develop*, dan *Deploy*. Setiap fase tersebut dikembangkan lebih lanjut dengan menambahkan sub-tahapan yang sesuai dengan konteks dan kebutuhan proyek agar proses pengembangan solusi AI menjadi lebih terstruktur dan sistematis.



Gambar 1. Ilustrasi Artificial Intelligence Life Cycle


TIN: Terapan Informatika Nusantara
Vol 99, No 99, Month Year, page 999-999
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/ tin.v99i99.9999

Adapun penjelasan rinci dari ilustrasi Pengembangan AI pada Gambar 1 diatas sebagai berikut:

a. *Design*
Fase *design* merupakan tahap awal dalam pengembangan sistem kecerdasan buatan yang berfokus pada pemahaman terhadap permasalahan, kebutuhan bisnis, dan identifikasi data yang relevan. Pada tahap ini, tim merancang arsitektur solusi, memilih algoritma yang sesuai, serta mempertimbangkan potensi risiko implementasi (De Silva & Alahakoon, 2022). Proses *design* mencakup pemahaman masalah secara menyeluruh, pengumpulan serta eksplorasi data dari berbagai sumber, hingga tahap pembersihan dan persiapan data melalui proses transformasi dan penggabungan agar dapat digunakan dalam pelatihan model.

b. *Develop*
Fase *develop* merupakan tahap yang berfokus pada eksplorasi, pelatihan model, serta evaluasi kinerja model menggunakan metrik yang sesuai (De Silva & Alahakoon, 2022). Pada fase ini, algoritma yang telah dipilih diimplementasikan dan diuji dalam berbagai skenario guna memastikan kemampuan model dalam melakukan generalisasi terhadap data baru. Tahap ini mencakup proses *modeling*, yaitu pemilihan, pelatihan, dan optimasi algoritma *machine learning* atau *deep learning* untuk menyelesaikan permasalahan yang telah diidentifikasi; serta *evaluation*, yaitu pengukuran performa model terhadap data yang belum pernah dilihat sebelumnya guna menilai keakuratan dan keandalannya.

c. *Deploy*
Fase *deploy* merupakan tahap di mana model AI yang telah dibangun dan diuji diintegrasikan ke dalam lingkungan produksi agar dapat digunakan oleh pengguna akhir atau sistem lain (De Silva & Alahakoon, 2022). Tantangan utama dalam fase ini meliputi performa *real-time*, pemantauan model setelah diterapkan, serta kebutuhan untuk melakukan retraining secara berkala ketika terjadi perubahan pada data dunia nyata (*data drift*). Tahapan ini mencakup proses *move to production*, yaitu penerapan model ke dalam sistem produksi agar dapat diakses dan digunakan secara nyata, serta *monitoring* and *maintenance*, yaitu kegiatan pemantauan kinerja dan pemeliharaan model guna memastikan bahwa model tetap akurat, andal, dan relevan dalam menghadapi kondisi operasional yang dinamis.

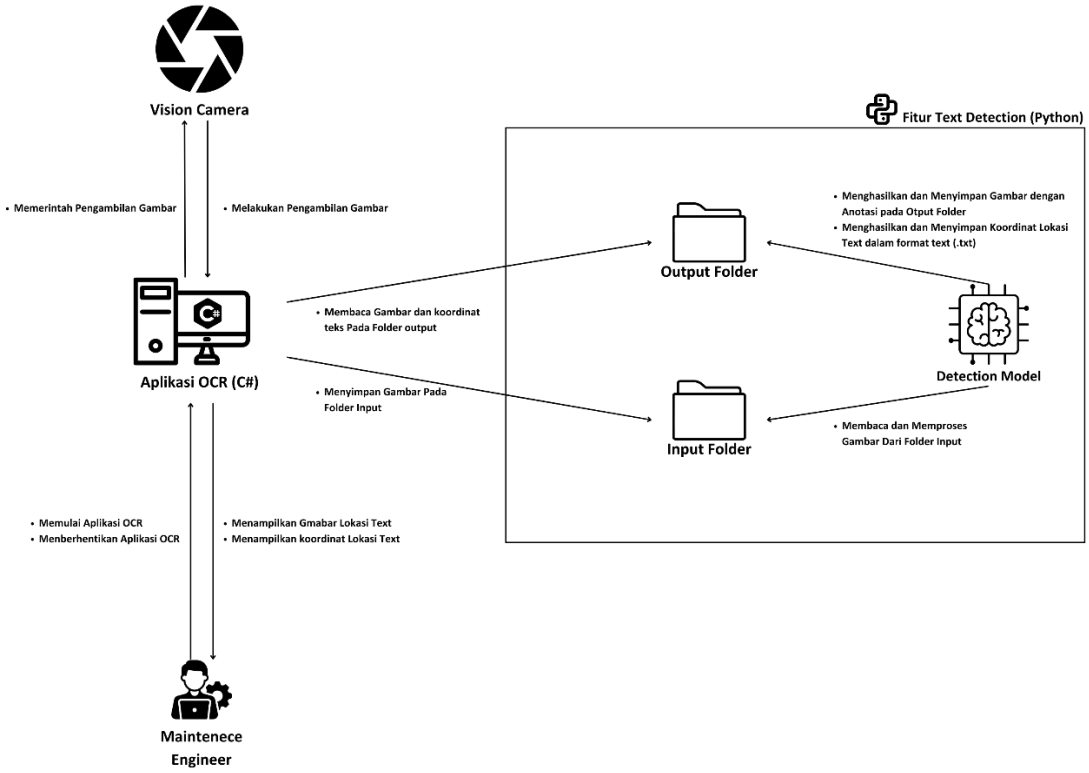
2.2 Gambaran Umum Sistem

Fitur *Text Detec*tion aplikasi OCR adalah fitur yang bertugas untuk mendeteksi teks dan menggambar *bounding box* pada area tersebut, dengan ada nya *bounding box* itu kita dapat mengetahui lokasi text pada gambar. Fitur *Text Detection* aplikasi OCR menggunakan algoritma YOLO yang terkenal dengan akurasi dan kecepatannya dalam mendeteksi objek. Dengan menggunakan algoritma YOLO diharapkan fitur Text Detection dapat menemukan lokasi dengan akurat dan cepat.

Sistem yang dikembangkan terdiri dari dua komponen utama aplikasi OCR utama yang sudah ada, yang berbasis C#, dan fitur Text Detection baru yang dikembangkan menggunakan Python dengan model YOLO. Fitur Text Detection ini berfungsi sebagai modul eksternal yang dipanggil oleh aplikasi C# untuk melakukan analisis gambar. Tujuannya adalah agar aplikasi OCR dapat secara otomatis menerima gambar, mengirimkannya ke fitur deteksi, dan kemudian menerima kembali gambar beranotasi beserta data koordinat lokasi teks.

Proses fitur *Text Detection* dimulai ketika *Maintenance Engineer* menjalankan aplikasi OCR. Aplikasi OCR kemudian mengirimkan perintah kepada *Vision Machine* untuk melakukan pengambilan gambar. Setelah gambar diambil, *Vision Machine* akan menyimpan gambar tersebut ke dalam direktori kamera. Selanjutnya, aplikasi OCR memindahkan gambar dari direktori kamera ke direktori input untuk diproses lebih lanjut. Fitur *Text Detection* secara otomatis akan memindai direktori input secara berkala setiap detik untuk mendeteksi gambar baru. Ketika gambar terdeteksi, fitur deteksi OCR meneruskannya ke model *Text Detection*, yang kemudian memproses gambar tersebut untuk mendeteksi keberadaan teks. Hasil dari proses ini berupa gambar yang telah di anotasi dengan *bounding box* dan file koordinat lokasi teks dalam format txt.

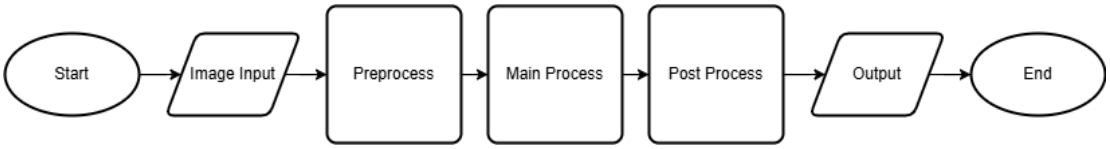
TIN: Terapan Informatika Nusantara
Vol 99, No 99, Month Year, page 999-999
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/ tin.v99i99.9999



Gambar 2. Gambaran Umum Sistem

Dari gambaran umum pada Gambar 2, dapat dilihat bahwa fitur *Text Detection* dalam aplikasi OCR memiliki beberapa fungsionalitas utama yang saling terhubung. Fitur ini secara otomatis akan aktif saat aplikasi OCR dijalankan oleh *Maintenance Engineer*, dan akan berhenti ketika aplikasi dihentikan. Selama aktif, fitur ini bertugas untuk memindai direktori input secara berkala, mendeteksi gambar baru yang masuk, dan memprosesnya menggunakan model YOLO untuk mendeteksi keberadaan teks. Proses ini menghasilkan dua jenis *output*, yaitu gambar yang telah di anotasi dengan *bounding box* dan file koordinat lokasi teks dalam format txt, yang kemudian dikembalikan ke aplikasi OCR. Fungsionalitas-fungsionalitas ini selanjutnya dijadikan acuan dalam proses pengujian sistem untuk memastikan bahwa setiap komponen bekerja sesuai dengan kebutuhan yang telah ditetapkan.

2.3 Flowchart Model



Gambar 3. Flowchart Model

Model *text Detection* yang telah dilatih menggunakan YOLO melewati beberapa tahap untuk dapat melakukan *detection* dan mengeluarkan *output*. Adapun tahapan-tahapan tersebut, sebagaimana ditunjukkan pada Gambar 3, adalah sebagai berikut:

a. *Preprocess*

Preprocessing merupakan tahap awal yang dilakukan untuk mempersiapkan data sebelum diproses oleh model. Pada *framework* seperti Ultralytics, proses ini berlangsung secara otomatis selama inferensi tanpa memerlukan campur tangan pengguna (Glenn Jocher, 2024). Salah satu langkah penting dalam tahap ini adalah *resize*, yaitu mengubah resolusi gambar ke ukuran yang diharapkan oleh model, misalnya dari 1280x720 menjadi 640x640 (Khan et al., 2022). Selanjutnya, dilakukan normalisasi nilai piksel dengan membaginya dengan 255 untuk menyederhanakan nilai numerik dan mempercepat konvergensi saat pelatihan atau inferensi. Penelitian menunjukkan bahwa normalisasi data input memberikan kontribusi signifikan terhadap kemampuan generalisasi model (Huang et al., 2023). Tahapan terakhir dalam *preprocess* adalah konversi gambar menjadi *tensor*, yaitu struktur data multidimensi yang memungkinkan pemrosesan numerik berskala besar pada CPU atau GPU. Tensor ini merupakan representasi numerik dari citra yang dibutuhkan agar model dapat memahami dan memproses informasi visual tersebut (A. Wang et al., 2024).

TIN: Terapan Informatika Nusantara
Vol 99, No 99, Month Year, page 999-999
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/ tin.v99i99.9999

b. *Main Process*
Main process adalah inti dari sistem deteksi objek, di mana model melakukan ekstraksi fitur, pemrosesan fitur multi-skala, dan prediksi terhadap objek yang terdeteksi. Komponen pertama adalah *backbone*, yang bertugas mengekstraksi ciri-ciri penting dari gambar dan mengubahnya menjadi *feature maps* yang merepresentasikan informasi visual dalam bentuk numerik (A. Wang et al., 2024). Setelah itu, informasi ini diproses oleh *neck*, yaitu bagian yang menggabungkan fitur dari berbagai skala dengan tujuan meningkatkan kemampuan model dalam mendeteksi objek dalam ukuran yang bervariasi. Neck sering kali mengimplementasikan struktur seperti *Feature Pyramid Network* (FPN) atau *Path Aggregation Network* (PANet) untuk memperkaya representasi fitur (Terven et al., 2023). Proses utama ditutup oleh *head*, yaitu komponen yang menghasilkan prediksi akhir. Head akan menentukan koordinat *bounding box*, *confidence score* yang menunjukkan tingkat keyakinan model terhadap keberadaan objek, serta *class probability* untuk mengidentifikasi kategori objek berdasarkan hasil deteksi (Setiyadi et al., 2023).

c. *Post Process*
Tahap *postprocess* bertujuan untuk menyempurnakan hasil prediksi mentah yang dihasilkan dari main process agar siap ditampilkan atau digunakan. Salah satu langkah penting dalam tahap ini adalah *confidence score filtering*, yang digunakan untuk menyaring *bounding box* dengan tingkat keyakinan rendah agar tidak disertakan dalam hasil akhir. Setelah itu, model menerapkan *Non-Maximum Suppression (NMS)*, sebuah metode yang berfungsi untuk mengeliminasi deteksi yang saling tumpang tindih dan mempertahankan hanya satu *bounding box* dengan *confidence score* tertinggi untuk setiap objek. Melalui tahap *postprocess* ini, model menghasilkan output yang lebih bersih dan akurat, baik dalam bentuk anotasi pada gambar maupun data lokasi teks dalam format teks seperti txt.

3. HASIL DAN PEMBAHASAN

3.1 *Design*

Pada fase *design* ada 3 langkah utama yang dilakukan yaitu pemahaman masalah (*Understand the Problem*), eksplorasi dan pengumpulan data (*Data Gathering and Exploration*), dan pembersihan dan persiapan data (*Data Wrangling and Preparation*).

3.1.1 Pemahaman Masalah (*Understand the Problem*)

Tahap awal dalam fase desain adalah memahami masalah bisnis secara mendalam. Permasalahan utama yang teridentifikasi di PT XYZ adalah proses quality control yang tidak efisien karena adanya kebutuhan untuk menandai lokasi teks pada gambar produk secara manual. Proses manual ini memakan waktu dan rentan terhadap kesalahan, terutama saat posisi teks pada produk sering berubah.

Solusi yang diusulkan adalah mengembangkan sebuah fitur deteksi teks otomatis menggunakan algoritma YOLO (You Only Look Once). Algoritma ini dipilih karena keunggulannya dalam kecepatan dan akurasi untuk deteksi objek berbasis gambar (image-based detection), yang sangat sesuai dengan kebutuhan sistem yang memproses gambar hasil tangkapan kamera. Meskipun proses OCR pada sistem ini tidak dilakukan secara *real-time* atau *live detection*, melainkan melalui pembacaan gambar hasil tangkapan kamera (*captured image*), algoritma YOLO tetap relevan karena tidak hanya kuat dalam *real-time detection*, tetapi juga memiliki performa yang sangat baik dalam *image-based detection* secara umum(Kosasi et al., 2024)

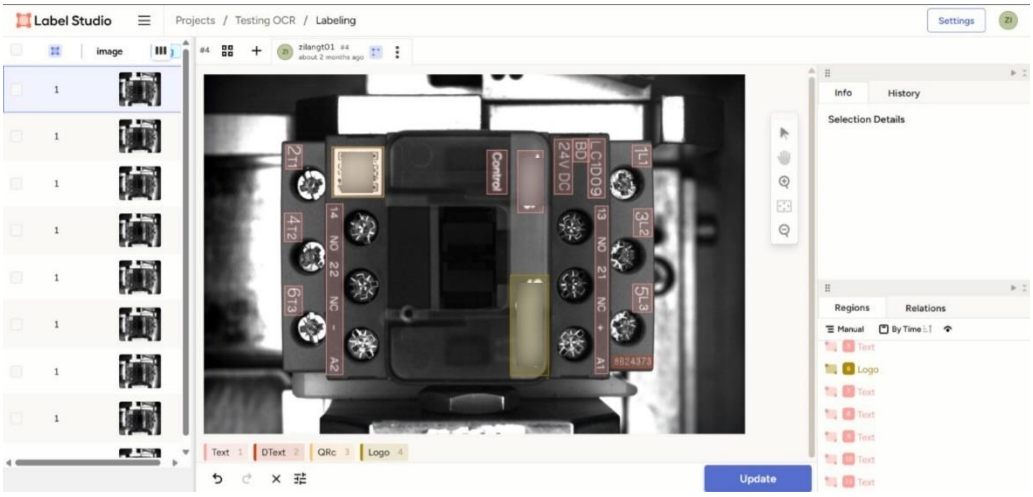
3.1.2 Eksplorasi dan Pengumpulan Data (*Data Gathering and Exploration*)

Untuk melatih model deteksi, langkah selanjutnya adalah mengumpulkan data gambar yang representatif. Bekerja sama dengan pengguna di PT XYZ, sebanyak 634 gambar dari 6 jenis produk industri yang berbeda berhasil dikumpulkan. Gambar-gambar ini diambil langsung dari mesin *vision camera* yang digunakan dalam proses produksi, sehingga data yang digunakan sangat relevan dengan kondisi operasional sehari-hari. Setiap gambar memiliki format *grayscale* dan memuat berbagai informasi teks yang nantinya akan menjadi target deteksi oleh model.

3.1.3 Pembersihan dan Persiapan Data (*Data Wrangling and Preparation*)

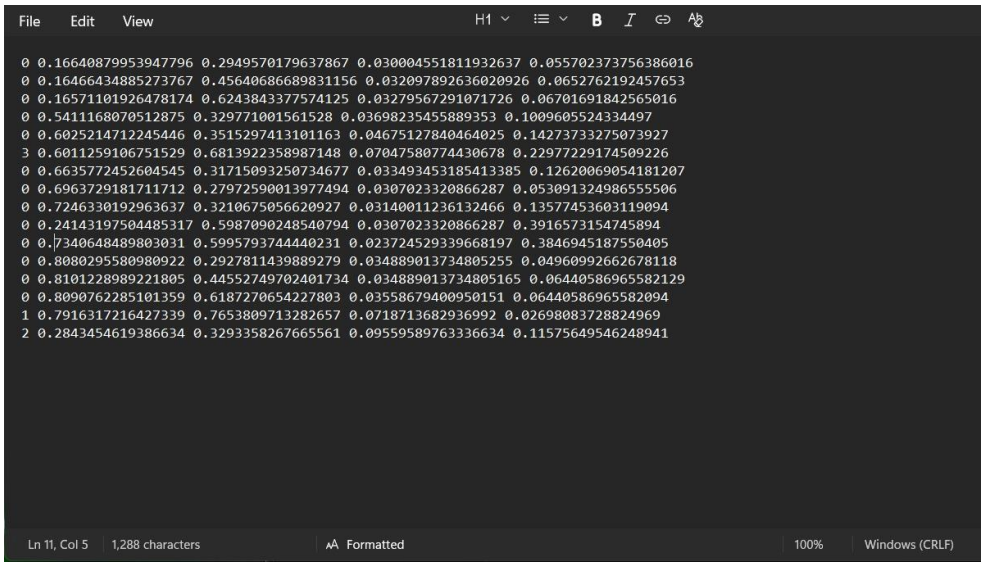
Setelah 634 gambar produk terkumpul, langkah selanjutnya adalah mengubah data mentah ini menjadi sebuah dataset yang terstruktur dan siap digunakan untuk melatih model. Tahapan ini mencakup pelabelan data dan pembagian dataset. Proses pelabelan (anotasi) dilakukan menggunakan *tool open-source* Label Studio. Pada tahap ini, setiap lokasi teks yang relevan pada gambar ditandai secara manual menggunakan *bounding box.* Proses ini sangat krusial karena kualitas label akan menentukan kemampuan model dalam mengenali teks secara akurat.

TIN: Terapan Informatika Nusantara
Vol 99, No 99, Month Year, page 999-999
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/ tin.v99i99.9999



Gambar 4. Labeling Text pada Gambar menggunakan Label Studio

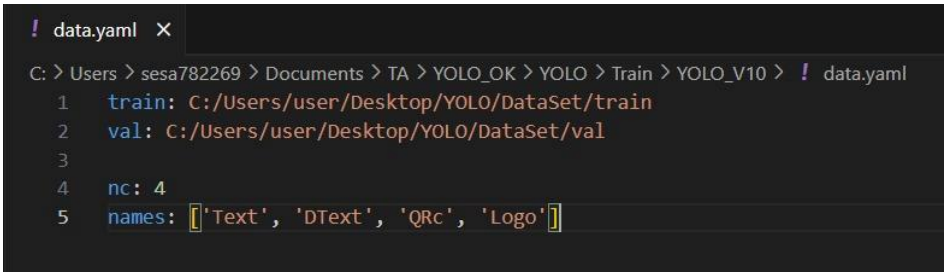
Berdasarkan Gambar 4 diatas adapun class atau label yang terdapat pada gambar adalah Text yang digunakan untuk area yang mengandung text, DText untuk menandai area yang mengandung text dengan orieantasi berbeda, QRc untuk menandai area QR code, dan Logo untuk menandai area yang mengandung logo produk. Setelah semua gambar dilabeli, anotasi diekspor ke dalam format YOLO. Format ini secara otomatis menghasilkan satu file .txt untuk setiap gambar, yang berisi informasi class dan koordinat dari setiap bounding box dalam format yang ternormalisasi (*x_center, y_center, width, height*) seperti pada Gambar 5 dibawah.



Gambar 5. File txt

Selanjutnya, dataset yang telah memiliki label dibagi menjadi dua set terpisah 80% sebagai data latih (*training set*), yang akan digunakan untuk melatih model. 20% sebagai data validasi (*validation set*), yang digunakan untuk mengevaluasi performa model pada data yang belum pernah ia lihat sebelumnya. Pemisahan dataset ini dilakukan secara terpisah untuk setiap jenis produk, pendekatan ini dilakukan dengan tujuan agar proporsi data setiap jenis produk tetap seimbang di antara data training dan evaluation, teknik ini dikenal sebagai *stratified sampling*, yang terbukti efektif dalam mengurangi bias bias terhadap kelas mayoritas (Lee & Ahn, 2023). Terakhir, file konfigurasi data.yaml dibuat untuk mendefinisikan path ke data latih/validasi, jumlah kelas dan nama-nama kelas yang dapat dilihat pada Gambar 6.

TIN: Terapan Informatika Nusantara
Vol 99, No 99, Month Year, page 999-999
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/ tin.v99i99.9999



Gambar 6s. File Data.yaml

3.2 Develop

Pada fase *develop* ada 2 langkah utama yang dilakukan yaitu pemilihan dan pelatihan model (*Modeling*), dan evaluasi (*Evaluation*).

3.2.1 Pemilihan dan Pelatihan Model (*Modeling*)

Pada fase pemilihan dan pelatihan model *(Modeling)*, dilakukan pemilihan jenis model yang mampu memberikan hasil terbaik. Setelah melakukan riset awal mengenai model YOLO, ditemukan bahwa YOLOv10 memiliki keunggulan dalam kecepatan inferensi dan akurasi deteksi, menjadikannya pilihan ideal untuk kebutuhan deteksi.

Berdasarkan penelitian oleh A. Wang et al. (2024), YOLOv10 menjadi salah satu model YOLO tercepat dan paling akurat pada masanya. Hal ini dicapai berkat penerapan fitur-fitur unggulan seperti *NMS-Free Training*, *Holistic Model Design*, dan *Enhanced Model Capabilities*.

Tabel 1. Tabel Perbandingan Kinerja YOLOv10

Model	Params (M)	FLOPs (G)	mAPval 50-95	Latency (ms)	Latency-forward (ms)
YOLOv6-3.0-N	4.7	11.4	37.0	2.69	1.76
Gold-YOLO-N	5.6	12.1	39.6	2.92	1.82
YOLOv8n	3.2	8.7	37.3	6.16	1.77
YOLOv10n	**2.3**	**6.7**	**39.5**	**1.84**	**1.79**
YOLOv6-3.0-S	18.5	45.3	44.3	3.42	2.35
Gold-YOLO-S	21.5	46.0	45.4	3.82	2.73
YOLOv8s	11.2	28.6	44.9	7.07	2.33
YOLOv10s	**7.2**	**21.6**	**46.8**	**2.49**	**2.39**
RT-DETR-R18	20.0	60.0	46.5	4.58	4.49
YOLOv6-3.0-M	34.9	85.8	49.1	5.63	4.56
Gold-YOLO-M	41.3	87.5	49.8	6.38	5.45
YOLOv8m	25.9	78.9	50.6	9.50	5.09
YOLOv10m	**15.4**	**59.1**	**51.3**	**4.74**	**4.63**
YOLOv6-3.0-L	59.6	150.7	51.8	9.02	7.90
Gold-YOLO-L	75.1	151.7	51.8	10.65	9.78
YOLOv8l	43.7	165.2	52.9	12.39	8.06

TIN: Terapan Informatika Nusantara
Vol 99, No 99, Month Year, page 999-999
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/ tin.v99i99.9999

Model	Params (M)	FLOPs (G)	mAPval 50-95	Latency (ms)	Latency-forward (ms)
RT-DETR-R50	42.0	136.0	53.1	9.20	9.07
YOLOv10l	**24.4**	**120.3**	**53.4**	**7.28**	**7.21**
YOLOv8x	68.2	257.8	53.9	16.86	12.83
RT-DETR-R101	76.0	259.0	54.3	13.71	13.58
YOLOv10x	**29.5**	**160.4**	**54.4**	**10.70**	**10.60**

Dari data pada Tabel 1 di atas, dapat dilihat bahwa varian YOLOv10 menunjukkan performa yang kompetitif pada *detection dataset* COCO, dengan kombinasi latensi rendah dan akurasi tinggi dibandingkan model-model sekelasnya. Tabel tersebut menyajikan perbandingan berbagai varian YOLOv10 terhadap model deteksi lain dalam konteks evaluasi pada dataset COCO, yang merupakan standar *benchmark* umum dalam pengujian sistem deteksi objek. Hasil ini menunjukkan bahwa YOLOv10 merupakan pilihan yang tepat untuk aplikasi yang menuntut efisiensi tinggi tanpa mengorbankan akurasi.

Setelah model YOLOv10 dipilih, proses pelatihan dimulai dengan memanfaatkan framework Ultralytics. Framework ini dikenal sangat memudahkan, baik untuk tahap pelatihan (training) maupun saat inferensi. Dalam pelatihan ini, beberapa parameter kunci ditetapkan, yaitu jumlah epoch sebanyak 100 dan ukuran gambar (image size) 640 piksel. Proses ini menggunakan dataset yang telah disiapkan dengan tujuan untuk menghasilkan model yang paling optimal dari sisi performa.

Untuk mengidentifikasi keseimbangan antara kecepatan inferensi dan akurasi, dilakukan pelatihan terpisah pada enam varian YOLOv10. Varian-varian tersebut dikategorikan berdasarkan skalanya yang mencakup n (*nano*), s (*small*), m (*medium*), b (*base*), l (*large*), dan x (*extra large*). Pemilihan varian ini secara langsung mempengaruhi dua hal utama, yaitu jumlah parameter dalam model dan kebutuhan sumber daya komputasi. Varian yang lebih kecil (seperti 'n') memiliki parameter lebih sedikit, sehingga menawarkan keunggulan dalam kecepatan dan efisiensi sumber daya komputasi, namun berpotensi mengorbankan akurasi. Sebaliknya, arsitektur yang lebih besar (seperti 'x') mengandung jumlah parameter yang signifikan, yang dirancang untuk mencapai akurasi maksimal namun dengan konsekuensi berupa kecepatan inferensi yang lebih lambat dan peningkatan kebutuhan sumber daya komputasi.

3.2.2 Evaluasi (*Evaluation*)

Langkah selanjutnya adalah *evaluation*, yaitu tahap di mana model yang telah dilatih dievaluasi untuk diukur performanya. Pada tahap ini, dilakukan pengujian terhadap data validasi dengan memerhatikan metrik umum yang biasa digunakan dalam evaluasi model deteksi seperti *mean Average Precision* (mAP), *precision*, *recall*, serta *inference time* (waktu prediksi).

Mean Average Precision (mAP) merupakan metrik utama yang digunakan dalam evaluasi performa model deteksi objek. Metrik ini memberikan gambaran menyeluruh tentang seberapa baik model dalam mengenali dan mengklasifikasikan berbagai jenis objek di dalam gambar. Evaluasi dengan mAP penting karena deteksi objek tidak hanya bergantung pada kemampuan mengenali objek secara akurat, tetapi juga pada seberapa konsisten model dalam mendeteksi semua objek yang relevan.

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

Precision adalah metrik yang digunakan untuk mengukur seberapa tepat model dalam membuat prediksi. Secara khusus, precision adalah proporsi dari prediksi positif yang benar terhadap seluruh prediksi positif yang dibuat model. Nilai precision yang tinggi menunjukkan bahwa model jarang membuat kesalahan berupa *false positives*, atau prediksi yang salah terhadap objek yang sebenarnya tidak ada.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Recall digunakan untuk mengukur seberapa banyak objek yang benar-benar ada dalam gambar berhasil dikenali oleh model. Ini menunjukkan kemampuan model dalam menghindari *false negatives*, yaitu kegagalan dalam mendeteksi objek yang sebenarnya ada. Nilai recall yang tinggi menandakan bahwa model mampu menemukan sebagian besar objek yang relevan, meskipun bisa saja precision-nya tidak terlalu tinggi jika model juga menghasilkan banyak deteksi palsu.

TIN: Terapan Informatika Nusantara
Vol 99, No 99, Month Year, page 999-999
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/ tin.v99i99.9999

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

F1 Score adalah metrik yang digunakan untuk menemukan keseimbangan antara Precision dan Recall. Metrik ini menggabungkan kedua metrik tersebut menjadi satu nilai tunggal, yang sangat berguna ketika melakukan penilaian kinerja model secara keseluruhan tanpa terlalu condong ke salah satu metrik saja. Nilai F1 Score yang tinggi menunjukkan bahwa model tersebut memiliki kinerja yang seimbang.

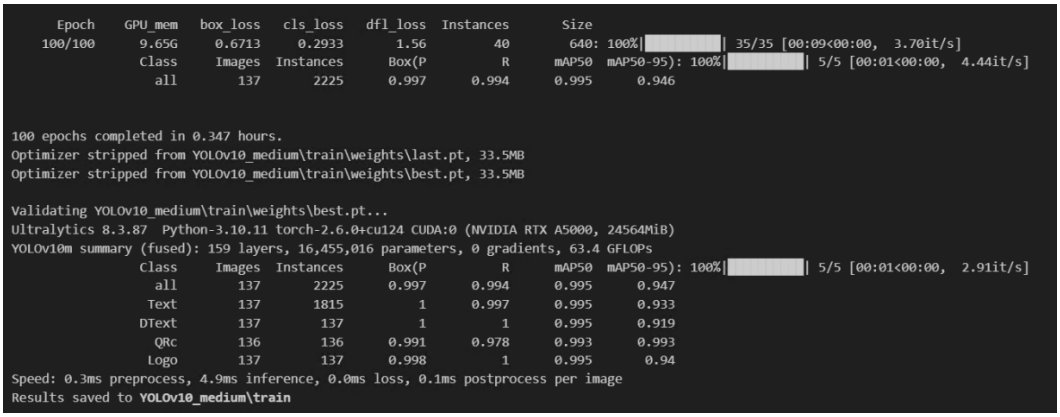
$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Average Speed (Kecepatan Rata-rata) digunakan untuk mengukur efisiensi dan kelayakan model untuk aplikasi praktis. Metrik ini menunjukkan total waktu rata-rata yang dibutuhkan model untuk memproses satu gambar yang mencakup preprocess, inference, postprocess. Nilai average speed yang rendah sangat penting untuk aplikasi real-time, di mana kecepatan respons model sama pentingnya dengan akurasi prediksinya.

Tabel 2. Tabel Evaluasi Model yang Dilatih

Model	F1	mAP@.50	mAP@.50-.95	Precision	Recall	Waktu Inferensi (ms)
YOLOv10n	0.980	0.993	0.939	0.986	0.974	3.4
YOLOv10s	0.986	0.994	0.946	0.984	0.988	4.0
YOLOv10m	**0.995**	**0.995**	**0.947**	**0.997**	**0.994**	**5.3**
YOLOv10b	0.995	0.995	0.945	0.995	0.996	5.9
YOLOv10l	0.997	0.995	0.947	0.997	0.997	7.2
YOLOv10x	0.994	0.995	0.950	0.990	0.999	8.5

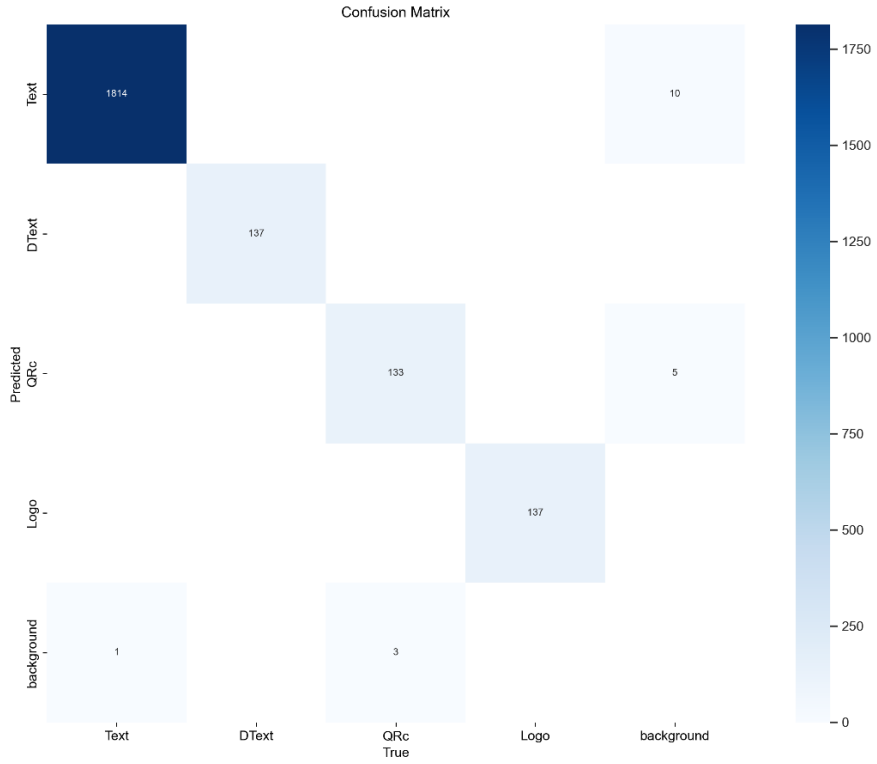
Berdasarkan hasil pelatihan dan evaluasi model yang ditampilkan pada Tabel 2, seluruh jenis model yang dilatih menunjukkan akurasi yang tinggi serta waktu inferensi yang sangat cepat. Namun, dari hasil evaluasi metrik seperti *precision*, *recall*, *f1* dan *mAP*, model YOLOv10 *medium* terbukti memiliki performa paling seimbang. Model ini mencatatkan nilai *precision*, *recall*, *F1* dan *mAP50* di atas 0,99, serta *mAP50-95* yang mencapai 0,946. Selain itu, precision dan recall untuk semua kelas juga berada di atas 0,99, menandakan bahwa model ini mampu mengenali objek dengan sangat baik di berbagai kelas.



Gambar 7. Hasil Evaluasi YOLOv10m

Efisiensi waktu inferensi juga menjadi keunggulan dari YOLOv10 *medium*. Seperti yang ditunjukkan pada Gambar 7, model ini memiliki rata-rata waktu inferensi hanya 5,3 ms, menjadikannya sangat ideal untuk aplikasi real-time tanpa mengorbankan akurasi. Waktu inferensi yang rendah ini menunjukkan bahwa YOLOv10 medium tidak hanya unggul dalam akurasi, tetapi juga dalam kecepatan dan efisiensi pemrosesan.

TIN: Terapan Informatika Nusantara
Vol 99, No 99, Month Year, page 999-999
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/ tin.v99i99.9999



Gambar 8. Confusion Matrix YOLOv10m

Berdasarkan hasil analisis *confusion matrix* pada Gambar 8, YOLOv10 *medium* menunjukkan konsistensi yang sangat baik dalam proses deteksi. Model ini menghasilkan jumlah false prediction paling rendah dibandingkan model lain, yang hanya terjadi pada dua kelas saja. Hal ini memperkuat kesimpulan bahwa YOLOv10 medium merupakan model dengan akurasi tertinggi dan kesalahan klasifikasi yang sangat minimal. Oleh karena itu, model ini dipilih sebagai model utama yang akan digunakan pada fitur deteksi.

3.3 Deploy

Pada fase deploy, model yang telah dievaluasi dan dipilih yaitu YOLOv10 Medium diintegrasikan ke dalam lingkungan produksi agar dapat digunakan secara fungsional oleh aplikasi OCR.

3.3.1 Langkah Implementasi (*Move to Production*)

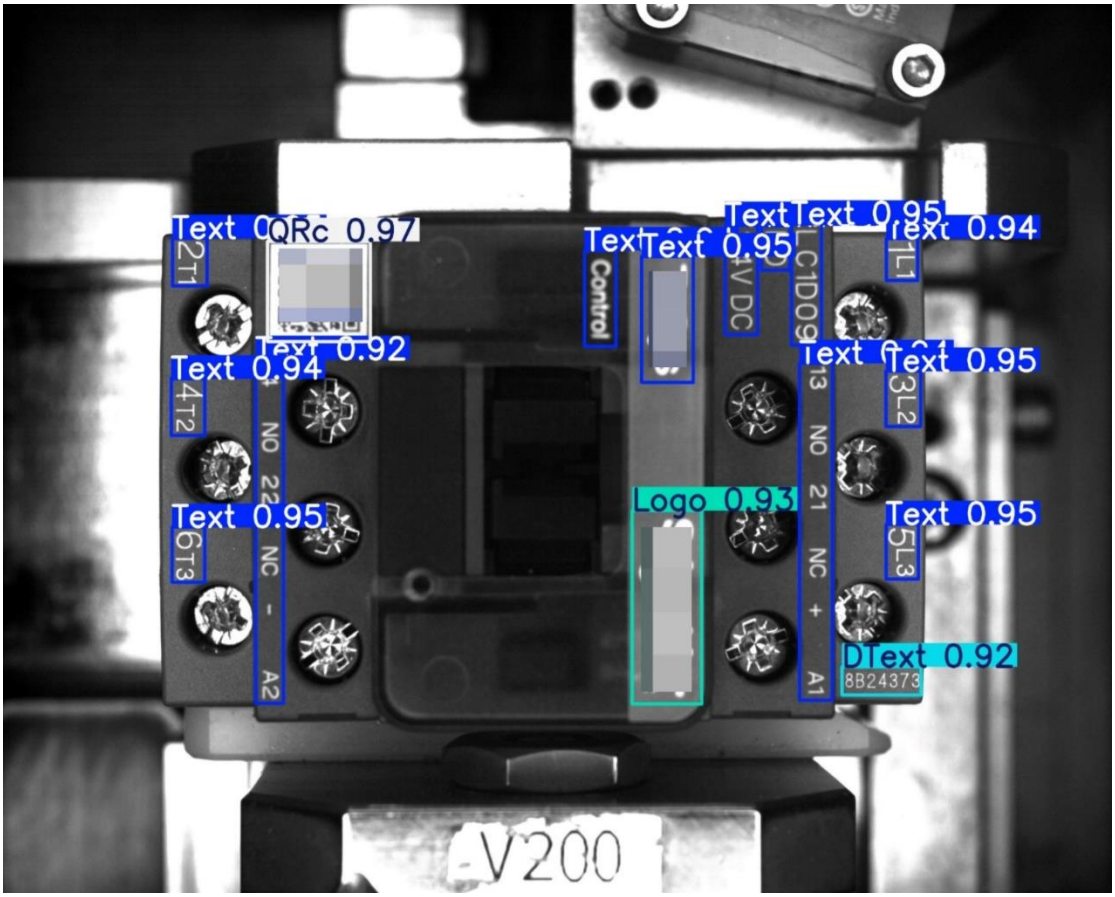
Model YOLOv10 *Medium* diimplementasikan menggunakan sebuah *script* Python yang memanfaatkan *framework* Ultralytics. Saat aplikasi OCR mengirimkan sebuah gambar untuk dideteksi, script ini akan menjalankan alur kerja inferensi yang berjalan sepenuhnya otomatis. Alur kerja ini merupakan inti dari cara model memproses gambar dan menghasilkan deteksi, yang terdiri dari tiga tahapan utama:

a. *Preprocess* (Prapemrosesan)
   Tahap ini secara otomatis mempersiapkan gambar input agar sesuai dengan format yang dibutuhkan oleh model. Proses ini mencakup beberapa langkah, yaitu mengubah resolusi gambar ke ukuran yang diharapkan model yaitu 640, normalisasi nilai piksel (biasanya dengan membaginya dengan 255) untuk mempercepat proses komputasi, dan konversi gambar menjadi format *tensor*, yaitu struktur data numerik yang dapat diproses secara efisien oleh GPU.
b. *Main Process* (Proses Utama)
   Ini adalah tahap di mana model melakukan deteksi. Tensor gambar yang telah disiapkan dilewatkan melalui tiga komponen arsitektur YOLO yaitu *Backbone* Mengekstrak fitur-fitur visual penting dari gambar dan mengubahnya menjadi *feature maps*. *Nec*k Menggabungkan feature maps dari berbagai skala untuk meningkatkan kemampuan model dalam mendeteksi objek dengan ukuran yang bervariasi. *Head* Menghasilkan prediksi akhir berupa koordinat *bounding box*, skor kepercayaan (*confidence score*), dan probabilitas kelas untuk setiap objek yang terdeteksi
c. *Postprocess* (Pasca-pemrosesan)

TIN: Terapan Informatika Nusantara
Vol 99, No 99, Month Year, page 999-999
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/ tin.v99i99.9999

Tahap terakhir ini bertugas untuk menyaring dan membersihkan hasil prediksi mentah dari model. Proses ini menerapkan penyaringan skor kepercayaan untuk membuang deteksi yang memiliki tingkat keyakinan rendah. Setelah itu, metode *Non Maximum Suppression* (NMS) digunakan untuk mengeliminasi bounding box yang tumpang-tindih dan hanya mempertahankan deteksi terbaik untuk setiap objek. Hasil akhirnya adalah *output* yang bersih berupa gambar beranotasi dan file koordinat teks.

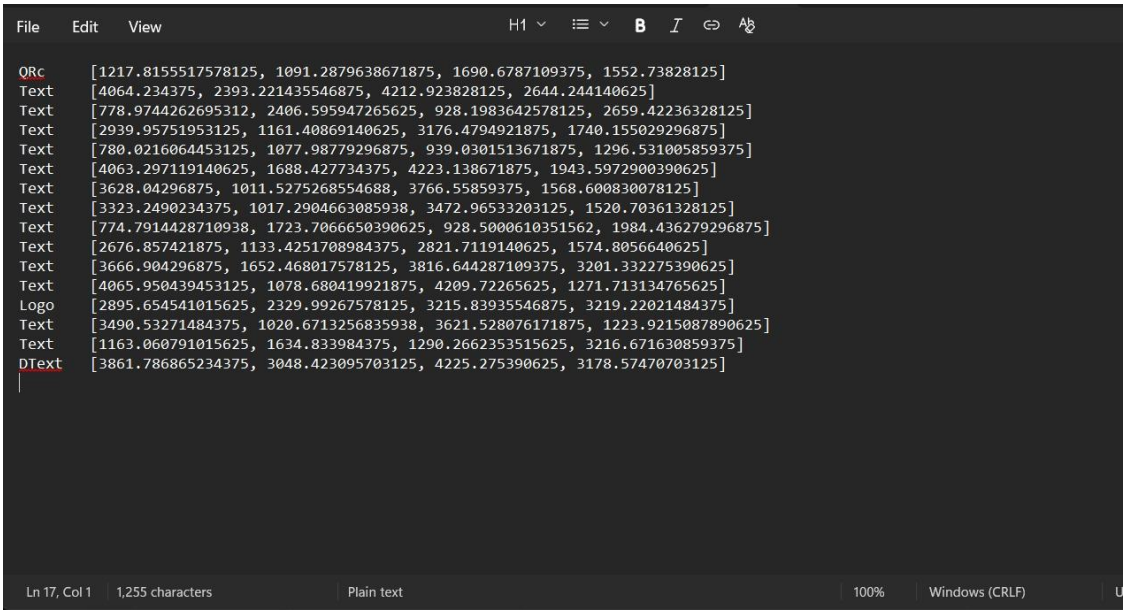
Script Python ini dirancang untuk secara aktif memindai (*scan*) sebuah folder *input*. Ketika gambar baru terdeteksi, gambar tersebut akan langsung diproses melalui alur kerja model yang terdiri dari tahap *preprocess*, *main process*, dan *postprocess*. Setelah pemrosesan selesai, script akan menghasilkan dua jenis *output* yaitu Gambar baru yang telah diberi anotasi *bounding box* untuk menandai lokasi dan sebuah file .txt yang berisi data koordinat dari setiap deteksi. Kedua file hasil ini kemudian disimpan secara otomatis ke dalam folder *output* yang telah ditentukan, siap untuk dibaca kembali oleh aplikasi OCR.



Gambar 9. Hasil Gambar dengan Anotasi

Gambar 9 memperlihatkan output visual dari *script* dalam bentuk gambar yang telah diberi anotasi bounding box. Bounding box ini menandai lokasi-lokasi di dalam gambar di mana teks telah berhasil dideteksi oleh model. Kotak-kotak tersebut biasanya berwarna mencolok untuk memudahkan identifikasi posisi teks secara langsung oleh pengguna. Gambar ini berfungsi sebagai bukti visual keberhasilan deteksi serta sebagai referensi bagi pengguna.

TIN: Terapan Informatika Nusantara
Vol 99, No 99, Month Year, page 999-999
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/ tin.v99i99.9999



Gambar 10. Hasil File txt dengan Koordinat text

Gambar 10 menunjukkan contoh dari file teks (.txt) yang dihasilkan bersamaan dengan gambar beranotasi. File ini berisi data koordinat setiap bounding box hasil deteksi dalam format yang terstruktur yaitu label dan koordinat lokasi text. Data ini sangat penting bagi sistem OCR karena data ini yang memungkinkan dilakukannya text recognition pada tahap selanjutnya pada aplikasi OCR.

Gambar 11. Aplikasi OCR(C#)

Untuk mengintegrasikan *script* Python tersebut ke dalam aplikasi OCR (C#), digunakan *System.Diagnostics.Process* sebagai mekanisme pemanggilan proses eksternal. Pendekatan ini memungkinkan aplikasi untuk menjalankan script Python seolah-olah dijalankan dari terminal, serta memungkinkan pembacaan output dan penanganan error dari proses Python secara langsung di dalam aplikasi seperti yang terlihata pada Gambar 11 diatas.

TIN: Terapan Informatika Nusantara
Vol 99, No 99, Month Year, page 999-999
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/ tin.v99i99.9999

3.4 Testing

Testing ini Bertujuan untuk memastikan bahwa fitur *text detection* yang telah dikembangkan dapat melakukan beberapa fungsi utama yang telah di paparkan pada gambaran umum sistem.

Tabel 3. Tabel Testing

No	Fungsi	Deskripsi	Luaran yang diharapkan	Status
1	Membaca dan memproses gambar dari folder	Sistem mendeteksi gambar baru yang masuk ke folder OCR dan memulai proses inferensi	Gambar terbaca dan diproses oleh fitur dteksi	Berhasil
2	Deteksi teks otomatis menggunakan YOLO	Sistem melakukan deteksi teks pada gambar menggunakan model YOLO dan menghasilkan output berupa bounding box	Bounding box ditampilkan pada gambar hasil, menunjukkan area teks dengan akurat	Berhasil
3	Menyimpan hasil deteksi sebagai gambar	Sistem menyimpan gambar hasil deteksi yang telah diberi anotasi ke folder output	File gambar anotasi tersimpan di folder output	Berhasil
4	Menyimpan koordinat label dalam file format txt	Sistem menyimpan hasil koordinat bounding box ke dalam file berformat txt	File txt dengan koordinat label tersimpan di folder output dan sesuai dengan anotasi gambar	Berhasil
5	Integrasi fitur text detection ke aplikasi OCR	Fitur text detection Python terhubung dengan aplikasi OCR berbasis C#	Aplikasi OCR dapat menjalankan dan menghentikan proses inferensi	Berhasil

4. KESIMPULAN

Penelitian ini bertujuan untuk mengatasi masalah inefisiensi dalam proses quality control di PT XYZ yang disebabkan oleh kebutuhan penandaan lokasi teks secara manual. Berdasarkan hasil implementasi dan pengujian, dapat disimpulkan bahwa tujuan penelitian telah tercapai. Pengembangan fitur *Text Detection* berbasis algoritma YOLOv10 Medium berhasil diintegrasikan ke dalam aplikasi OCR. Sistem ini secara efektif mengotomatiskan proses deteksi teks, yang sebelumnya merupakan pekerjaan manual yang memakan waktu. Model yang dihasilkan menunjukkan performa yang sangat andal dengan mAP@50-95 sebesar 0.947 serta nilai *precisio, recall*dan *F1* di atas 0.99. Pengujian fungsional telah memvalidasi bahwa seluruh alur kerja, mulai dari pemrosesan gambar otomatis hingga penyimpanan hasil berfungsi dengan benar. Dengan demikian, produk yang dihasilkan secara langsung menyelesaikan masalah yang diidentifikasi di awal. Namun, masih terdapat ruang untuk perbaikan. Nilai Box Loss sebesar 0.6713 menunjukkan bahwa akurasi posisi bounding box masih dapat ditingkatkan. Saran untuk pengembangan selanjutnya adalah menambah jumlah dan variasi data pelatihan serta menerapkan teknik augmentasi data untuk meningkatkan ketangguhan dan presisi model.

REFERENCES

Cai, B. (2023). Deep learning Optical Character Recognition in PCB Dark Silk Recognition. *World Journal of Engineering and Technology*, *11*(01), 1–9. https://doi.org/10.4236/wjet.2023.111001

De Silva, D., & Alahakoon, D. (2022). An artificial intelligence life cycle: From conception to production. *Patterns*, *3*(6), 100489. https://doi.org/10.1016/j.patter.2022.100489

Glenn Jocher. (2024, May 24). *YOLOv10: Real-Time End-to-End Object Detection*. Https://Docs.Ultralytics.Com/Models/Yolov10/.

TIN: Terapan Informatika Nusantara
Vol 99, No 99, Month Year, page 999-999
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/ tin.v99i99.9999

Haifeng, D., & Siqi, H. (2020). Natural scene text detection based on YOLO V2 network model. *Journal of Physics: Conference Series*, *1634*(1), 012013. https://doi.org/10.1088/1742-6596/1634/1/012013

Huang, L., Qin, J., Zhou, Y., Zhu, F., Liu, L., & Shao, L. (2023). Normalization Techniques in Training DNNs: Methodology, Analysis and Application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(8), 10173–10196. https://doi.org/10.1109/TPAMI.2023.3250241

Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2022). Transformers in Vision: A Survey. *ACM Computing Surveys*, *54*(10s), 1–41. https://doi.org/10.1145/3505244

Kosasi, T., Sihombing, Z. A. L., & Husein, A. M. (2024). YOLO-Based Vehicle Detection: Literature Review. *Journal of Computer Networks, Architecture and High Performance Computing*, *6*(3), 1384–1389. https://doi.org/10.47709/cnahpc.v6i3.4377

Lee, H., & Ahn, S. (2023). Improving the Performance of Object Detection by Preserving Balanced Class Distribution. *Mathematics*, *11*(21), 4460. https://doi.org/10.3390/math11214460

Maleh, I. M. D., Teguh, R., Sahay, A. S., Okta, S., & Pratama, M. P. (2023). Implementasi Algoritma You Only Look Once (YOLO) Untuk Object Detection Sarang Orang Utan Di Taman Nasional Sebangau. *Jurnal Informatika*, *10*(1), 19–27. https://doi.org/10.31294/inf.v10i1.13922

Mamuriyah, N., & Jacky, J. (2021). Perancangan dan Pembuatan Alat untuk Mendeteksi Teks Hangul dan Inggris pada Menu Makanan Menggunakan metode OCR (Optical Character Recognition). *Telcomatics*, *6*(1), 1–10. https://doi.org/10.37253/telcomatics.v6i1.5054

Marcellino Jonathan, Muhammad Thoriqul Hafidz, Nur Ayu Apriyanti, Zaid Husaini, & Perani Rosyani. (2023). MENDETEKSI PLAT NOMOR KENDARAAN DENGAN METODE YOLO (You Only Look Once) DAN SINGLE SHOT DETECTOR (SSD). *AI Dan SPK : Jurnal Artificial Intelligent Dan Sistem Penunjang Keputusan*, *1*, 105–111. https://jurnalmahasiswa.com/index.php/aidanspk/article/view/320

Palanikumar, K., Natarajan, E., & Ponshanmugakumar, A. (2024). Application of machine vision technology in manufacturing industries—a study. In *Machine Intelligence in Mechanical Engineering* (pp. 91–122). Elsevier. https://doi.org/10.1016/B978-0-443-18644-8.00018-6

Ramadhan, G., Khairiyah, R. D. A., Natania, S., & Harris, A. (2024). Vehicle Police Number Detection Using Yolov8. *Media Journal of General Computer Science*, *1*(2), 62–70. https://doi.org/10.62205/mjgcs.v1i2.25

Setiyadi, A., Utami, E., & Ariatmanto, D. (2023). Analisa Kemampuan Algoritma YOLOv8 Dalam Deteksi Objek Manusia Dengan Metode Modifikasi Arsitektur. In *Jurnal Sains Komputer & Informatika (J-SAKTI* (Vol. 7, Issue 2).

Terven, J., Córdova-Esparza, D.-M., & Romero-González, J.-A. (2023). A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. *Machine Learning and Knowledge Extraction*, *5*(4), 1680–1716. https://doi.org/10.3390/make5040083

Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., & Ding, G. (2024). YOLOv10: Real-Time End-to-End Object Detection. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, & C. Zhang (Eds.), *Advances in Neural Information Processing Systems* (Vol. 37, pp. 107984–108011). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2024/file/c34ddd05eb089991f06f3c5dc36836e0-Paper-Conference.pdf

Wang, X., Zheng, S., Zhang, C., Li, R., & Gui, L. (2021). R-YOLO: A Real-Time Text Detector for Natural Scenes with Arbitrary Rotation. *Sensors*, *21*(3), 888. https://doi.org/10.3390/s21030888