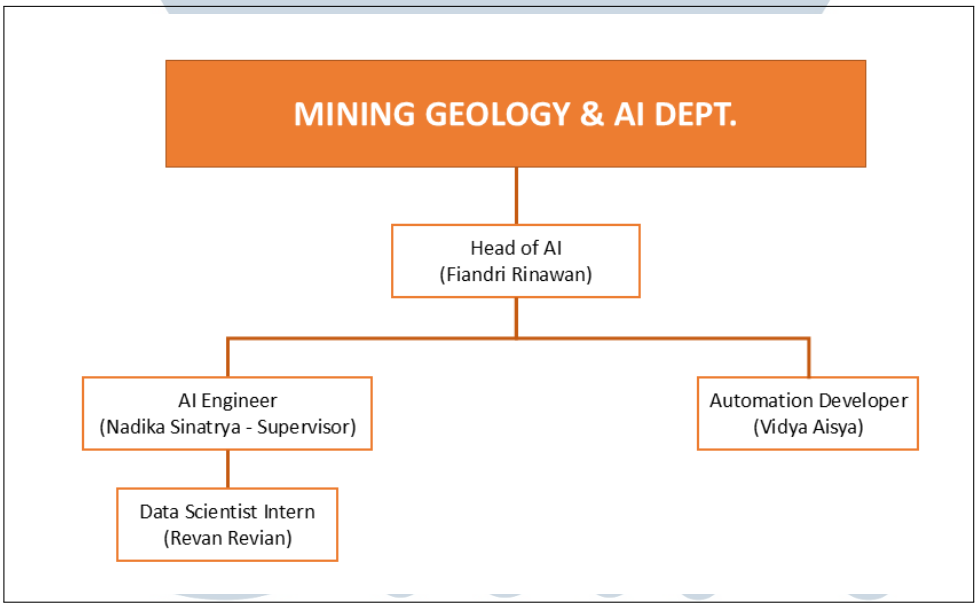
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Penugasan kerja magang ditempatkan pada divisi CKBE, tepatnya di *Mining Geology &* AI *Department*, dengan peran sebagai *Data Scientist Intern*. Struktur organisasi unit tersebut ditampilkan pada Gambar 3.1 dan menunjukkan posisi *intern* dalam hierarki serta jalur koordinasi antarpersonel. Dalam kedudukan ini, pemberian arahan dilaksanakan langsung oleh *supervisor* perusahaan yang berperan sebagai AI *Engineer*, sedangkan mekanisme koordinasi dilakukan secara berkala untuk memastikan setiap penugasan selaras dengan ketentuan dan standar kerja yang berlaku.



Gambar 3.1. Struktur organisasi *Mining Geology &* AI *Department* pada divisi CKBE

Pola kerja bersifat *hybrid*. Koordinasi jarak jauh dilaksanakan melalui Zoom, baik untuk komunikasi teks maupun rapat video. Kegiatan pemrograman dan analisis data menggunakan Google Colab sebagai lingkungan kerja utama. Selain itu, koordinasi tatap muka juga dijalankan secara periodik di kantor Jakarta pada saat pelaksanaan WFO. Pada tahap pengembangan, Visual Studio Code dimanfaatkan sebagai lingkungan pengembangan terintegrasi dan Docker Desktop digunakan untuk kontainerisasi layanan pendukung. Pemanfaatan kedua alat tersebut meningkatkan konsistensi lingkungan eksekusi dan memudahkan replikasi

proses saat kolaborasi tim.

3.2 Tugas yang Dilakukan

Selama pelaksanaan kerja magang, tanggung jawab utama meliputi kegiatan analisis data, pengembangan model *machine learning*, serta evaluasi hasil untuk mendukung pengambilan keputusan berbasis data di lingkungan perusahaan. Secara umum, proses kerja difokuskan pada dua proyek utama, yaitu:

1. Pengembangan sistem RAG berbasis LLM untuk analisis berita dan dokumen internal,
2. Pengembangan sistem OMR untuk deteksi *bubble* pada formulir inspeksi alat berat.

3.3 Uraian Pelaksanaan Magang

Tabel 3.1. Aktivitas Mingguan Selama Magang

Minggu Ke-	Pekerjaan yang dilakukan
1	Mempelajari konsep dasar LLM, RAG, serta teknik *document retrieval* menggunakan LangChain sebagai dasar pengembangan sistem AI berbasis teks.
2	Mempelajari penggunaan n8n sebagai alat automasi *workflow*, serta melakukan instalasi Docker Desktop dan integrasi n8n untuk keperluan pengembangan sistem.
3	Mengimplementasikan proses *data ingestion* pada sistem RAG yang mencakup *document processing*, *embedding*, serta eksplorasi model AI yang sesuai untuk sistem tanya jawab berbasis teks.
4	Mengembangkan model LLM dan *chatbot* berbasis RAG agar mampu memahami tabel dan daftar *bullet points*, serta menghasilkan *output* dalam format CSV untuk mendukung analisis data perusahaan.
5	Melakukan *debugging*, serta mengevaluasi model *embedding* dengan kualitas representasi teks yang lebih baik.

Tabel 3.1 Aktivitas Mingguan Selama Magang (lanjutan)

Minggu Ke-	Pekerjaan yang dilakukan
6–7	Mengembangkan antarmuka Gradio untuk *upload* file dan visualisasi hasil RAG, serta menyusun Excel dan ERD berdasarkan formulir *Machine Time Log Sheet* dan *Pre Start Check* sebagai persiapan proyek OMR.
8–9	Mengembangkan *pipeline* OMR untuk deteksi *bubble* menggunakan model YOLO dan HuggingFace, serta merancang data *synthetic* untuk keperluan *training*.
10–11	Melakukan perbaikan dan optimasi *pipeline* OMR, termasuk penyesuaian data *synthetic*, penerapan *guardrails* untuk mencegah salah deteksi, serta proses *fine-tuning* menggunakan data asli.
12–13	Menambahkan variasi *synthetic data* dan *pixel-based guardrails*, menyesuaikan hasil per pertanyaan agar selaras antara visualisasi dan CSV, serta meningkatkan ketepatan hasil deteksi berdasarkan inspeksi visual.
14–15	Mengembangkan antarmuka sistem OMR, mengembangkan *frontend* dan *backend*, melakukan *dockerization* terhadap layanan, serta memperbaiki sejumlah *bug* yang ditemukan selama pengujian.
16	Melakukan normalisasi basis data dan penyempurnaan akhir sistem agar siap digunakan dalam tahap evaluasi *internal* tim.

3.4 Pengembangan Sistem LLM Berbasis RAG untuk Ekstraksi Informasi dari Dokumen

3.4.1 *User Requirements*

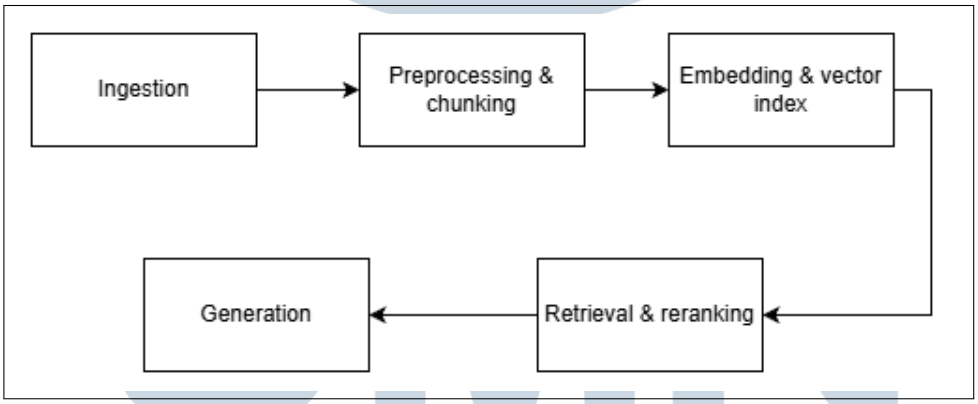
Untuk meningkatkan efisiensi dalam proses pengolahan dokumen *internal*, divisi CKBE menetapkan sejumlah kebutuhan fungsional terhadap sistem yang akan dikembangkan. Sistem diharapkan mampu mengekstraksi informasi penting dari berbagai dokumen perusahaan yang berbentuk teks, seperti notulen rapat dan berita industri. Hasil ekstraksi tersebut akan digunakan untuk mendukung analisis operasional serta pengambilan keputusan strategis berbasis data.

Secara umum, sistem harus mampu melakukan beberapa fungsi utama, yaitu pencarian berbasis konteks menggunakan LLM, penyajian jawaban yang relevan melalui mekanisme RAG, serta pengelolaan sumber data referensi yang dapat diperbarui secara dinamis. Melalui pendekatan tersebut, sistem diharapkan dapat menjawab pertanyaan pengguna berdasarkan isi dokumen internal maupun artikel eksternal dengan hasil yang akurat dan kontekstual.

Selain itu, sistem perlu dilengkapi dengan antarmuka berbasis *web* yang mudah digunakan, memungkinkan pengguna untuk *upload* dokumen baru, menelusuri hasil ekstraksi, serta memverifikasi kebenaran jawaban yang dihasilkan. Fitur pencarian dan penarikan data harus memiliki waktu respons yang singkat serta mendukung pengolahan dalam bahasa Indonesia dan Inggris.

3.4.2 Perancangan Sistem

Arsitektur konseptual dari sistem RAG yang dikembangkan ditunjukkan pada Gambar 3.2, yang menggambarkan alur mulai dari proses *upload* dokumen hingga generasi jawaban oleh model.



Gambar 3.2. *Pipeline* sistem RAG untuk ekstraksi informasi dari dokumen internal

3.4.3 Implementasi

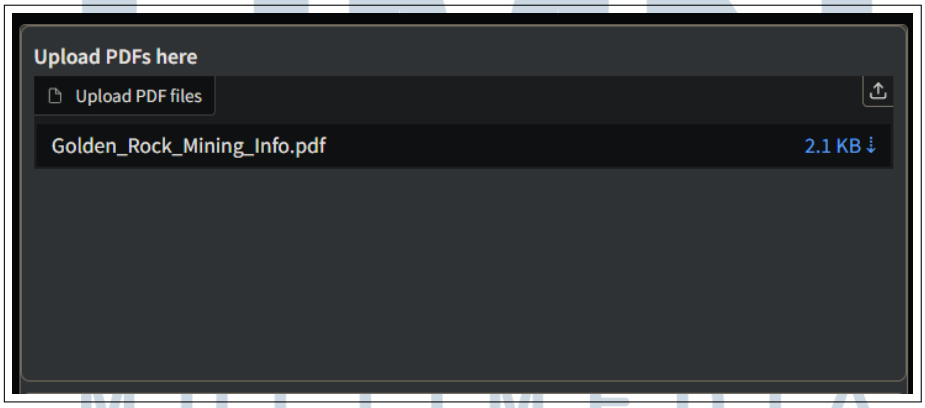
A *Ingestion*

Tahapan *ingestion* berperan sebagai pintu awal dalam proses ekstraksi informasi berbasis dokumen digital. Pada sistem ini, seluruh dokumen yang akan diproses di-*upload* dalam format PDF melalui antarmuka pengguna berbasis *web*. Setelah dokumen di-*upload*, sistem akan secara otomatis mendeteksi dan memproses setiap *file* yang tersedia di direktori khusus penyimpanan.

Untuk memastikan seluruh konten dokumen dapat diekstraksi secara maksimal, proses *parsing* dilakukan menggunakan dua pendekatan. Pertama, sistem memanfaatkan *PyPDFLoader* untuk mengambil teks utama dari setiap halaman PDF. Pendekatan ini efektif dalam menangkap isi narasi dan penjelasan yang umumnya terdapat pada laporan, notulen, atau dokumen internal perusahaan. Kedua, sistem juga menjalankan ekstraksi tabel menggunakan *pdfplumber*, yang secara khusus dirancang untuk membaca dan mengonversi tabel menjadi format CSV. Hasil ekstraksi tabel ini tetap dijaga keutuhannya agar struktur data tidak rusak dan mudah digunakan dalam proses analitik berikutnya.

Setiap konten yang berhasil diekstraksi, baik berupa teks maupun tabel, akan dibungkus ke dalam objek *Document* lengkap dengan *metadata*. Informasi tambahan seperti nama *file*, nomor halaman, serta indeks tabel turut dicatat untuk memudahkan proses pelacakan dan referensi di tahap selanjutnya. Pendekatan ini membuat seluruh bagian dokumen dapat diproses lebih lanjut secara terstruktur, termasuk mendukung pencarian berbasis konteks dan rekonstruksi sumber informasi secara presisi.

Seluruh proses *ingestion* ini berjalan secara otomatis di belakang layar, sehingga pengguna tidak perlu melakukan konfigurasi tambahan. Hasilnya, dokumen yang di*upload* dapat langsung terintegrasi ke dalam sistem, siap untuk dianalisis dan diolah lebih lanjut melalui pipeline RAG. Tampilan fitur *upload* dokumen diperlihatkan pada Gambar 3.3.



Gambar 3.3. Tampilan antarmuka untuk *upload* dokumen PDF pada tahap *ingestion*

B *Preprocessing & chunking*

Setelah dokumen berhasil di-*upload* dan diekstrak, langkah berikutnya adalah melakukan *preprocessing* agar data siap diolah lebih lanjut oleh sistem. Pada

tahap ini, seluruh teks dan tabel yang sudah diekstrak dikumpulkan dan dibersihkan dari elemen-elemen yang tidak diperlukan, seperti spasi kosong atau format yang tidak konsisten. Setiap potongan dokumen akan dilengkapi *metadata* penting, misalnya nama *file*, nomor halaman, hingga informasi jenis konten (apakah berupa teks biasa atau tabel).

Sistem ini mengadopsi pendekatan *chunking* yang fleksibel. Untuk dokumen yang memiliki struktur seperti *heading* atau *bullet* yang menyerupai format *markdown*, proses pemotongan dilakukan dengan mengandalkan kombinasi antara *MarkdownHeaderTextSplitter* dan *RecursiveCharacterTextSplitter*. Tujuannya, setiap *chunk* yang dihasilkan tetap mempertahankan konteks aslinya, misalnya satu subjudul dan paragraf yang saling berkaitan tidak terpotong secara sembarangan. Namun untuk bagian tabel, sistem secara sengaja tidak melakukan *chunking*, supaya seluruh isi tabel tetap utuh dan tidak kehilangan struktur baris kolomnya.

Jika dokumen tidak memiliki struktur yang jelas atau minim *heading*, maka sistem otomatis beralih ke pemotongan berbasis karakter. Setiap bagian dokumen dipotong per beberapa ratus karakter, dilengkapi *overlap* secukupnya supaya konteks tidak hilang. Hasil akhirnya, seluruh isi dokumen baik teks naratif, daftar, maupun tabel akan berubah menjadi kumpulan *chunk* kecil yang siap diolah oleh tahap berikutnya.

Tahapan ini memastikan data yang masuk ke sistem siap diproses secara efisien, baik untuk kebutuhan pencarian maupun analitik berbasis dokumen. Dengan strategi *chunking* yang adaptif dan pemberian *metadata* yang lengkap, proses pencarian dan penarikan informasi pada tahap berikutnya menjadi lebih akurat dan terarah, serta siap digunakan dalam *pipeline* analitik keseluruhan.

C *Embedding & vector index*

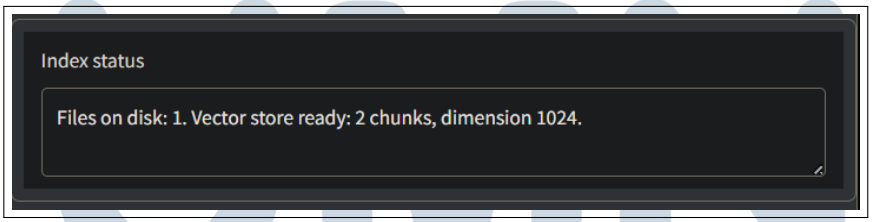
Setelah *chunking* selesai, setiap *chunk* diubah menjadi vektor numerik menggunakan *SentenceTransformer* dengan model *Qwen/Qwen3-Embedding-0.6B*. Proses *encode* berjalan per *batch* kecil agar stabil di memori dan menampilkan *progress bar* untuk pemantauan. *output* disatukan ke dalam *NumPy array* bertipe *float32*, sehingga siap dipakai oleh mesin pencari vektor di tahap berikutnya.

Pengembangan *vector index* dilakukan memakai *FAISS* tipe *IndexFlatL2*. Dimensi ruang vektor diturunkan langsung dari bentuk *embedding*, lalu seluruh vektor ditambahkan ke indeks. Indeks yang sudah terbangun disimpan ke *file faiss_index.idx* agar bisa dipakai ulang tanpa perlu mengembangkan dari awal.

Mekanisme ini diikat ke alur Gradio sehingga setiap kali pengguna menambah atau menghapus *file* PDF, sistem menjalankan ulang proses pemuatan, *embedding*, dan penulisan indeks. Terdapat *guard* sederhana yang memastikan model *embedding* dan fungsi pembuat *embedding* telah tersedia sebelum proses dimulai, serta menangani kondisi ketika tidak ada konten PDF yang valid.

Untuk pengambilan konteks, *query* pengguna terlebih dahulu diubah menjadi vektor menggunakan model yang sama, lalu dilakukan pencarian kesamaan vektor pada indeks *FAISS* guna mendapatkan calon *chunk* paling relevan dalam jumlah yang lebih besar dari target. Daftar calon ini kemudian dinilai ulang oleh *reranker* berbasis *CrossEncoder* dari JinaAI agar urutan relevansi lebih presisi, terutama pada kasus teks yang mirip secara semantik tetapi berbeda konteks. Hasil akhirnya adalah kumpulan *chunk* terbaik yang akan diteruskan ke tahap penyusunan jawaban.

Dari sisi operasional, seluruh langkah dirancang agar efisien tetapi tetap transparan. Penyimpanan indeks di disk mempercepat *warm start*, *float32* menjaga keseimbangan antara akurasi dan jejak memori, serta pemilihan perangkat untuk *reranker* menyesuaikan ketersediaan CUDA atau CPU. Dengan susunan ini, sistem memperoleh kombinasi kecepatan pencarian, ketepatan pemeringkatan, dan kemudahan perawatan dalam *pipeline* RAG secara keseluruhan. Tampilan status akhir proses ini ditunjukkan pada Gambar 3.4.



Gambar 3.4. Status *vector store* setelah proses *embedding* dan pengembangan indeks *FAISS* berhasil dilakukan

D *Retrieval & reranking*

Tahap ini dimulai dengan proses *vector search* menggunakan indeks *FAISS*. Model yang digunakan adalah *IndexFlatL2*, yaitu tipe indeks sederhana berbasis perhitungan jarak Euclidean (*L2 distance*) antara vektor *query* dan seluruh vektor dokumen yang disimpan. Pendekatan ini tidak menggunakan struktur hierarki atau approximate search, sehingga hasil pencarian yang diperoleh bersifat presisi penuh, meskipun dengan beban komputasi yang sedikit lebih tinggi dibanding metode yang

lebih kompleks.

Setelah kandidat terkumpul, urutan relevansi diperbaiki dengan *cross-encoder* JinaAI, dengan model *jinaai/jina-reranker-v2-base-multilingual*. Berbeda dari *similarity search* yang hanya melihat jarak vektor, *reranker* membaca pasangan *[query, chunk]* secara utuh lalu memberi skor semantik yang lebih tajam. Pemrosesan berjalan di CUDA jika tersedia atau otomatis turun ke CPU. Hasilnya, sistem mendapatkan *top-k chunk* yang benar-benar paling relevan dengan maksud *query*, bukan sekadar yang paling dekat secara vektor.

Pemilihan parameter dibuat adaptif. Untuk *query* yang terdeteksi berhubungan dengan tabel atau CSV, sistem menaikkan *k* dan menerapkan *window* pendek agar struktur baris kolom tidak terpecah. Di luar kasus tersebut, *k* disetel moderat agar latensi tetap rendah tanpa mengorbankan kualitas konteks. Praktik ini membantu menyeimbangkan presisi jawaban dan kecepatan respon, terutama ketika dokumen panjang dan bercampur antara narasi dan tabel.

Chunk yang lolos kemudian disusun kembali menjadi blok konteks lengkap beserta metadata sumber *file* untuk dimasukkan ke *prompt*. Aturan ketat pada *system prompt* memastikan jawaban hanya memakai informasi dari konteks yang diambil serta selalu menyertakan sitasi nama *file* dan nomor *chunk*. Dengan alur *retrieve* lalu *rerank* ini, sistem menjaga dua hal sekaligus: cakupan informasi yang cukup lebar di awal, lalu pemilahan yang presisi di akhir sebelum jawaban dihasilkan.

E *Generation*

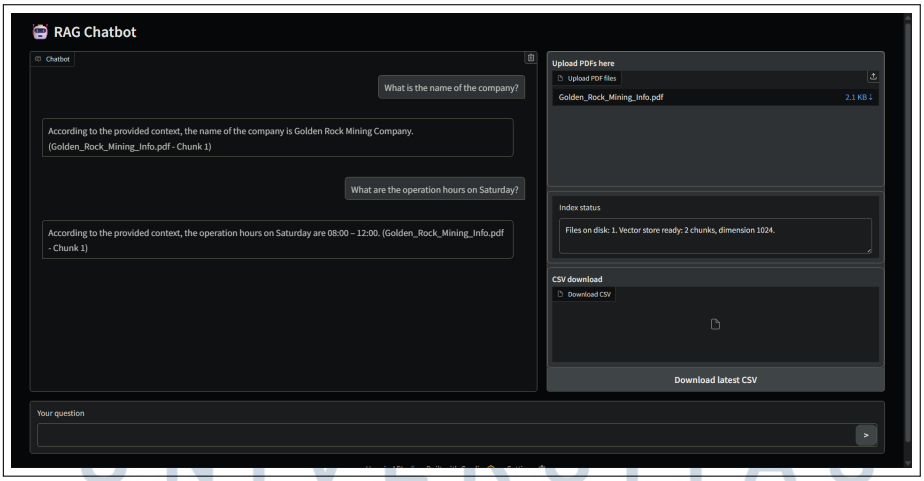
Setelah sistem mengambil dan menyusun konteks yang paling relevan, model LLM *Meta Llama 3 8B Instruct* digunakan untuk membuat jawaban dengan mengikuti aturan ketat yang sudah ditentukan di *system prompt*. Selama proses ini, jumlah percakapan yang dimasukkan dibatasi berdasarkan anggaran token agar respons tetap ringkas dan tidak melebihi batas memori. Jika proses berjalan terlalu berat atau gagal menghasilkan *output* karena keterbatasan resource, sistem akan otomatis mencoba dengan jumlah token yang lebih kecil sampai jawaban bisa diberikan.

Setiap jawaban yang dihasilkan dicek ulang agar tetap sesuai dengan konteks yang sudah diambil. Model hanya boleh menggunakan data yang benar-benar ada di konteks, dan wajib mencantumkan sumber berupa nama *file* dan nomor *chunk*. Jika informasi yang diminta pengguna tidak ditemukan dalam konteks, sistem akan

secara otomatis membalas dengan kalimat penolakan yang sudah ditetapkan. Hal ini bertujuan untuk menghindari munculnya jawaban halusinasi dan menjaga kejelasan asal informasi.

Untuk jawaban berupa tabel atau data CSV, ada proses tambahan supaya formatnya benar-benar sesuai dan bisa langsung dipakai. Sistem akan mencari blok CSV baik yang terdeteksi otomatis dari kode atau yang memiliki jumlah kolom konsisten. Setelah itu, baris-baris data dinormalisasi supaya setiap baris punya jumlah kolom sama seperti header, sel kosong tetap dipertahankan, dan jika ada baris yang kurang satu kolom pertama akan diberi padding. Jika sel pertama kosong tapi sebelumnya sudah ada nilainya, maka nilai terakhir akan diisi otomatis agar data tetap rapi. Sebelum *file* CSV ini disediakan untuk diunduh, sistem memastikan minimal ada nilai numerik di dalamnya agar tidak terjadi error atau salah simpan.

Terakhir, proses integrasi ke antarmuka Gradio memungkinkan pengguna langsung mengunduh *file* CSV hasil ekstraksi yang lolos validasi. Selain itu, riwayat percakapan tetap diperbarui secara ringkas, dan bila anggaran token hampir penuh, sistem secara otomatis membuat ringkasan agar percakapan tetap bisa dilanjutkan dengan konteks yang cukup. Tampilan antarmuka chatbot yang memuat seluruh proses ini ditunjukkan pada Gambar 3.5.



Gambar 3.5. Antarmuka utama chatbot berbasis RAG pada tahap *generation post-processing*

3.4.4 Evaluasi dan Pengajuan Hasil Model

Pada tahap ini, validasi hasil sistem RAG dilakukan melalui pemeriksaan kualitatif oleh pengguna untuk memastikan kesesuaian jawaban terhadap konteks dokumen sumber. Pendekatan ini dipilih karena *output* sistem berupa jawaban

terbuka berbasis teks, sehingga evaluasi lebih relevan dilakukan melalui verifikasi ketercakupan konteks, ketepatan sitasi sumber, serta kelayakan format CSV yang dihasilkan.

Hasil evaluasi tersebut kemudian diajukan kepada *supervisor* sebagai bagian dari proses *review* dan penyelarasan kebutuhan fitur. Pengajuan dilakukan melalui sesi demonstrasi sistem dengan skenario uji yang merepresentasikan kebutuhan pengguna, mencakup proses *upload* dokumen, pengajuan pertanyaan, pemeriksaan sitasi sumber, serta pengunduhan *output* CSV. Pada sesi ini, *supervisor* menilai kesesuaian jawaban terhadap dokumen rujukan dan memberikan *feedback* terhadap aspek fungsional maupun pengalaman pengguna, seperti struktur jawaban, konsistensi sitasi, dan stabilitas format CSV. *Feedback* yang diperoleh digunakan sebagai dasar iterasi pengembangan berikutnya hingga sistem dinilai telah memenuhi kebutuhan pada tahap *proof of concept* dan siap untuk dilanjutkan ke tahapan integrasi serta evaluasi lanjutan.

3.5 Pengembangan Sistem *Optical Mark Recognition* untuk Deteksi *Bubble* pada Formulir Hasil *Scan*

3.5.1 *User Requirements*

Pengembangan *Optical Mark Recognition* (OMR) ditujukan untuk menyediakan proses identifikasi *bubble* pada formulir hasil *scan* secara otomatis dan konsisten. Sistem diharapkan mampu membaca berbagai format masukan (*input*), termasuk berkas berjenis *image* dan PDF, kemudian mengonversinya menjadi citra beresolusi tinggi yang siap diproses. Kebutuhan ini meliputi kemampuan sistem dalam menangani variasi kualitas pemindaian, seperti rotasi ringan, pencahayaan tidak merata, serta perbedaan perangkat pemindai yang digunakan.

Untuk tahap deteksi, sistem membutuhkan komponen pendeteksi *bubble* berbasis *object detection* yang dilatih menggunakan *synthetic dataset*. Penggunaan *synthetic dataset* diperlukan karena ketersediaan data hasil *scan* asli masih terbatas, sehingga model tetap dapat belajar bentuk *ring bubble* dan variasi pola isian sejak awal. *Object detection* tersebut harus mampu mengenali *ring bubble* pada 57 baris pertanyaan, memetakan setiap kandidat *bubble* ke posisi kolom kiri atau kanan berdasarkan letak geometrisnya, lalu menghasilkan koordinat awal sebagai dasar tahap klasifikasi. Untuk memenuhi kebutuhan akurasi, proses deteksi harus tetap

stabil meskipun citra mengandung gangguan seperti garis tabel, *noise* sensor, atau variasi tinta pada lembar pemeriksaan.

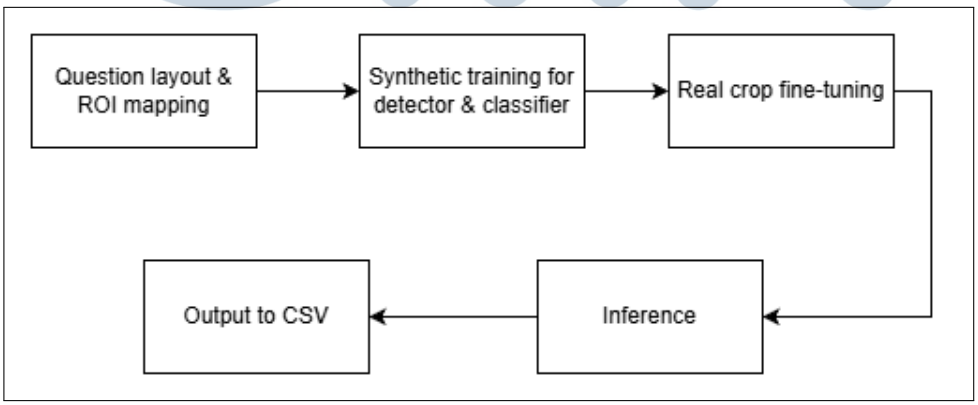
Setelah kandidat *bubble* teridentifikasi, sistem memerlukan modul klasifikasi berbasis *patch-level CNN* untuk membedakan kondisi *filled* dan *empty*. Klasifikasi harus mempertimbangkan beragam bentuk isian, termasuk titik kecil, coretan, tanda silang, maupun isian penuh yang muncul pada *bubble*. Agar dapat digunakan pada kondisi nyata, model membutuhkan proses *fine-tuning* menggunakan *real scanned crops* yang merepresentasikan pola isian aktual operator lapangan. Hasil *output* klasifikasi harus memuat probabilitas keyakinan untuk mendukung proses pengambilan keputusan pada level baris pertanyaan.

Selain itu, sistem harus mampu melakukan penentuan label akhir per baris berdasarkan dua kolom (kiri dan kanan), dengan kategorisasi Baik, Tidak Baik, atau Kosong. Mekanisme ini mencakup penetapan ambang probabilitas dan penanganan kasus *bubble* ganda. Sebagai *output*, seluruh hasil yang telah diproses kemudian disusun dalam format CSV yang mencakup nama *file*, pertanyaan, opsi terpilih, tingkat keyakinan, serta koordinat *bounding box* untuk setiap *bubble* yang terpilih.

Output sistem juga perlu menyediakan fungsi visualisasi untuk mendukung proses verifikasi. Fitur ini memerlukan pewarnaan kotak deteksi berdasarkan kategori label dan penempatan teks keterangan secara jelas pada citra asli, sehingga hasil prediksi dapat dibaca dengan cepat oleh pemeriksa.

3.5.2 Perancangan Sistem

Arsitektur dari sistem OMR yang dikembangkan ditunjukkan pada Gambar 3.6, yang menggambarkan hubungan antara proses *row decoding*, deteksi, klasifikasi, dan penyusunan CSV sebagai *output* akhir.

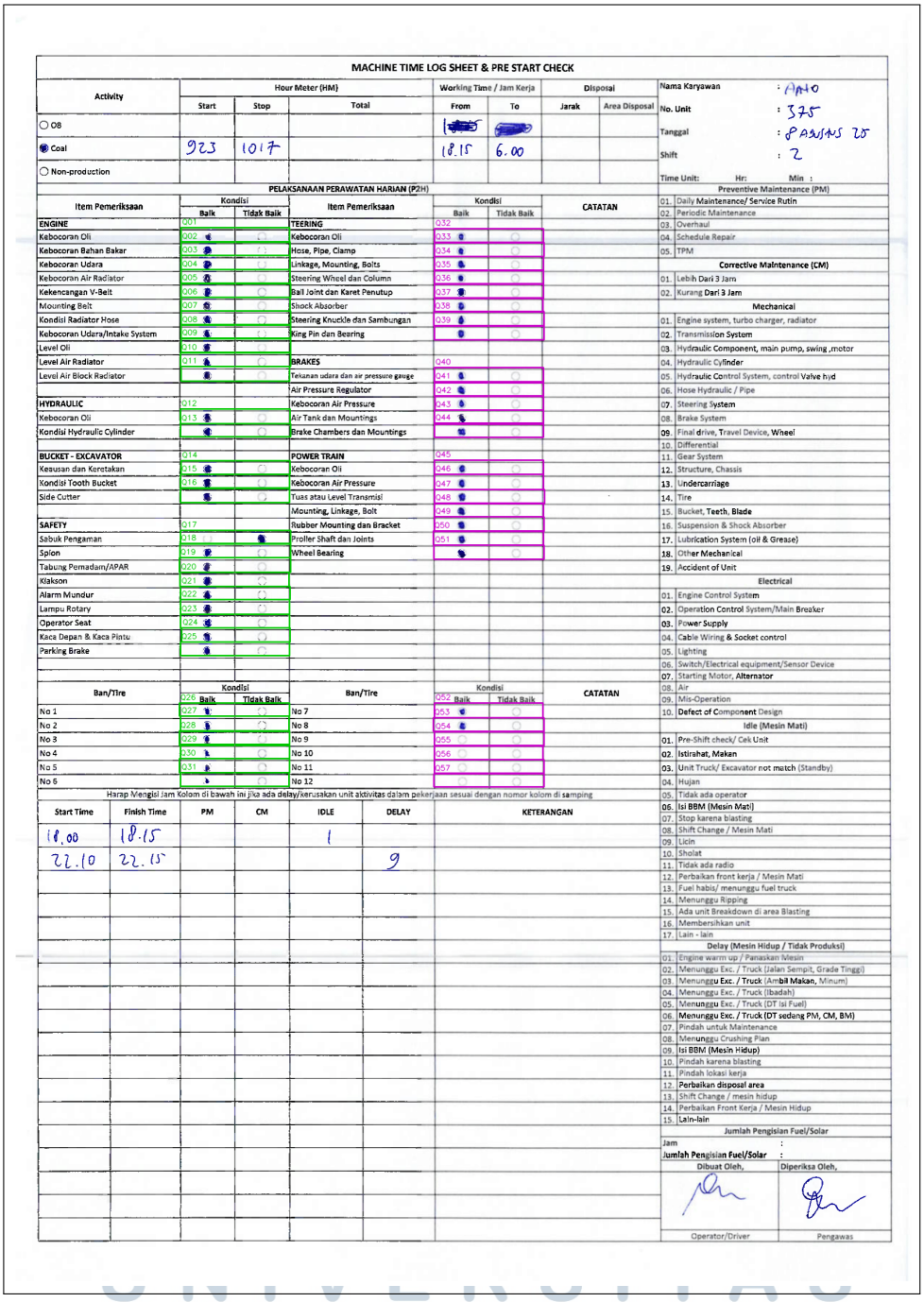


Gambar 3.6. *Pipeline* arsitektur OMR yang digunakan untuk deteksi dan klasifikasi *bubble* pada formulir hasil *scan*

3.5.3 Implementasi

A *Question layout & ROI mapping*

Tahap ini menjadi dasar utama dari seluruh alur OMR karena proses deteksi hanya dapat bekerja dengan baik apabila posisi setiap baris pertanyaan dipetakan secara konsisten. Formulir inspeksi terdiri dari 57 baris pertanyaan yang dibagi menjadi dua blok utama, yaitu blok kiri dan blok kanan. Setiap baris memiliki struktur visual yang tetap berupa sepasang *bubble* untuk pilihan Baik dan Tidak Baik. Gambaran umum mengenai pembagian baris dan area yang diproses sistem ditunjukkan pada Gambar 3.7.



MACHINE TIME LOG SHEET & PRE START CHECK

Activity	Hour Meter (HM)			Working Time / Jam Kerja		Disposal	
	Start	Stop	Total	From	To	Jarak	Area Disposal
○ OB							
● Coal	923	1017		18.15	6.00		
○ Non-production							

Nama Karyawan : ARIO
No. Unit : 375
Tanggal : 8 AGUSTUS 25
Shift : 2
Time Unit: Hr: Min :

PELAKSANAAN PERAWATAN HARIAN (P2H)

Item Pemeriksaan	Kondisi Baik	Kondisi Tidak Baik	Item Pemeriksaan	Kondisi Baik	Kondisi Tidak Baik	CATATAN
ENGINE	Q01		STEERING	Q32		
Kebocoran Oli	Q02		Kebocoran Oli	Q33		
Kebocoran Bahan Bakar	Q03		Hose, Pipe, Clamp	Q34		
Kebocoran Udara	Q04		Linkage, Mounting, Bolts	Q35		
Kebocoran Air Radiator	Q05		Steering Wheel dan Column	Q36		
Kekencangan V-Belt	Q06		Ball Joint dan Karet Penutup	Q37		
Mounting Belt	Q07		Shock Absorber	Q38		
Kondisi Radiator Hose	Q08		Steering Knuckle dan Sambungan	Q39		
Kebocoran Udara/Intake System	Q09		King Pin dan Bearing			
Level Oli	Q10					
Level Air Radiator	Q11		BRAKES	Q40		
Level Air Block Radiator			Tekanan udara dan air pressure gauge	Q41		
			Air Pressure Regulator	Q42		
HYDRAULIC	Q12		Kebocoran Air Pressure	Q43		
Kebocoran Oli	Q13		Air Tank dan Mountings	Q44		
Kondisi Hydraulic Cylinder			Brake Chambers dan Mountings			
BUCKET - EXCAVATOR	Q14		POWER TRAIN	Q45		
Keausan dan Keretakan	Q15		Kebocoran Oli	Q46		
Kondisi Tooth Bucket	Q16		Kebocoran Air Pressure	Q47		
Side Cutter			Tuas atau Level Transmisi	Q48		
			Mounting, Linkage, Bolt	Q49		
SAFETY	Q17		Rubber Mounting dan Bracket	Q50		
Sabuk Pengaman	Q18		Proller Shaft dan Joints	Q51		
Spion	Q19		Wheel Bearing			
Tabung Pemadam/APAR	Q20					
Klakson	Q21					
Alarm Mundur	Q22					
Lampu Rotary	Q23					
Operator Seat	Q24					
Kaca Depan & Kaca Pintu	Q25					
Parking Brake						

Ban/Tire	Kondisi Baik	Kondisi Tidak Baik	Ban/Tire	Kondisi Baik	Kondisi Tidak Baik	CATATAN
	Q26			Q52		
No 1	Q27		No 7	Q53		
No 2	Q28		No 8	Q54		
No 3	Q29		No 9	Q55		
No 4	Q30		No 10	Q56		
No 5	Q31		No 11	Q57		
No 6			No 12			

Harap Mengisi Jam Kolom di bawah ini jika ada delay/kerusakan unit aktivitas dalam pekerjaan sesuai dengan nomor kolom di samping

Start Time	Finish Time	PM	CM	IDLE	DELAY	KETERANGAN
18.00	18.15			1		
22.10	22.15				9	

Preventive Maintenance (PM)
01. Daily Maintenance/ Service Rutin
02. Periodic Maintenance
03. Overhaul
04. Schedule Repair
05. TPM

Corrective Maintenance (CM)
01. Lebih Dari 3 Jam
02. Kurang Dari 3 Jam

Mechanical
01. Engine system, turbo charger, radiator
02. Transmission System
03. Hydraulic Component, main pump, swing ,motor
04. Hydraulic Cylinder
05. Hydraulic Control System, control Valve hyd
06. Hose Hydraulic / Pipe
07. Steering System
08. Brake System
09. Final drive, Travel Device, Wheel
10. Differential
11. Gear System
12. Structure, Chassis
13. Undercarriage
14. Tire
15. Bucket, Teeth, Blade
16. Suspension & Shock Absorber
17. Lubrication System (oil & Grease)
18. Other Mechanical
19. Accident of Unit

Electrical
01. Engine Control System
02. Operation Control System/Main Breaker
03. Power Supply
04. Cable Wiring & Socket control
05. Lighting
06. Switch/Electrical equipment/Sensor Device
07. Starting Motor, Alternator
08. Air
09. Mis-Operation
10. Defect of Component Design

Idle (Mesin Mati)
01. Pre-Shift check/ Cek Unit
02. Istirahat, Makan
03. Unit Truck/ Excavator not match (Standby)
04. Hujan
05. Tidak ada operator
06. Isi BBM (Mesin Mati)
07. Stop karena blasting
08. Shift Change / Mesin Mati
09. Licin
10. Sholat
11. Tidak ada radio
12. Perbaikan front kerja / Mesin Mati
13. Fuel habis/ menunggu fuel truck
14. Menunggu Ripping
15. Ada unit Breakdown di area Blasting
16. Membersihkan unit
17. Lain - lain

Delay (Mesin Hidup / Tidak Produksi)
01. Engine warm up / Panaskan Mesin
02. Menunggu Exc. / Truck (Jalan Sempit, Grade Tinggi)
03. Menunggu Exc. / Truck (Ambil Makan, Minum)
04. Menunggu Exc. / Truck (Ibadah)
05. Menunggu Exc. / Truck (DT Isi Fuel)
06. Menunggu Exc. / Truck (DT sedang PM, CM, BM)
07. Pindah untuk Maintenance
08. Menunggu Crushing Plan
09. Isi BBM (Mesin Hidup)
10. Pindah karena blasting
11. Pindah lokasi kerja
12. Perbaikan disposal area
13. Shift Change / mesin hidup
14. Perbaikan Front Kerja / Mesin Hidup
15. Lain-lain

Jumlah Pengisian Fuel/Solar
Jam :
Jumlah Pengisian Fuel/Solar :

Dibuat Oleh, Diperiksa Oleh,

Operator/Driver | Pengawas

Gambar 3.7. Ilustrasi pembagian baris dan *region of interest* (ROI) untuk setiap pertanyaan pada formulir inspeksi.

Untuk setiap baris, sistem menetapkan *region of interest* yang mencakup area pertanyaan beserta dua *bubble* di sekitarnya. ROI ini digunakan untuk memotong citra menjadi potongan kecil per baris sebelum dikirim ke model *object detection*. Dengan cara ini, model hanya melihat area yang relevan untuk satu baris tertentu sehingga gangguan dari elemen lain seperti garis tabel atau teks bagian lain

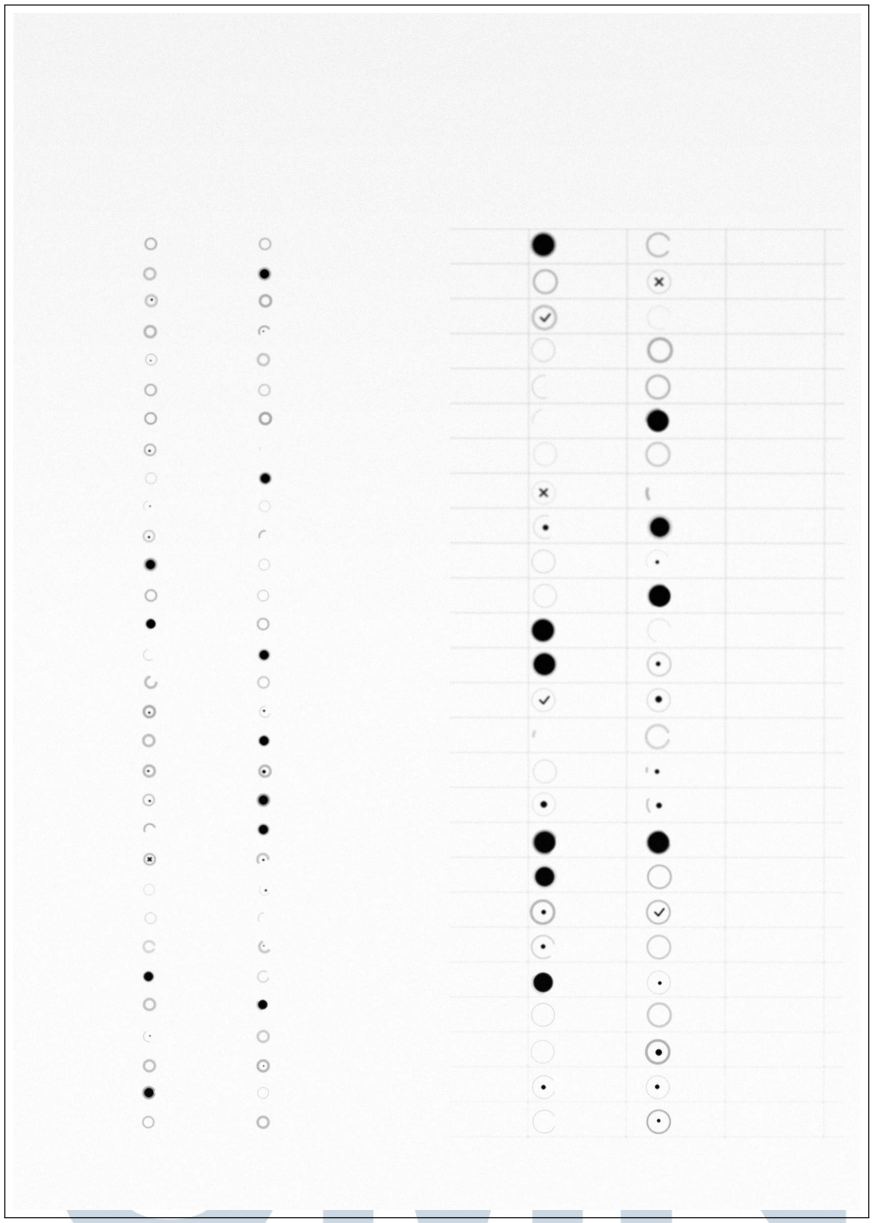
dapat dikurangi. Di dalam setiap ROI, sistem juga menyimpan posisi pusat kolom kiri dan kanan sehingga kandidat *bubble* dapat ditentukan sisi mana yang paling sesuai.

Agar layout tetap berfungsi untuk berbagai ukuran dan resolusi hasil *scan*, seluruh koordinat ROI disimpan dalam bentuk ternormalisasi terhadap lebar dan tinggi halaman. Pada proses *inference*, nilai ternormalisasi tersebut dikonversi kembali menjadi koordinat piksel sehingga lokasi setiap baris tetap konsisten meskipun dokumen berasal dari pemindai yang berbeda. Pendekatan ini membantu sistem mengatasi variasi seperti sedikit rotasi, perbedaan *zoom*, maupun rasio aspek yang tidak persis sama.

Struktur layout yang sama juga digunakan pada tahap pembuatan *synthetic dataset*. Halaman sintetis digambar menggunakan konfigurasi baris dan posisi kolom *bubble* yang identik dengan formulir asli sehingga model detektor dan *patch-level classifier* belajar bahwa setiap indeks pertanyaan selalu terkait dengan lokasi fisik tertentu pada halaman. Konsistensi ini membuat *pipeline* mampu menggabungkan hasil deteksi dan klasifikasi menjadi label akhir yang stabil untuk setiap baris pertanyaan.

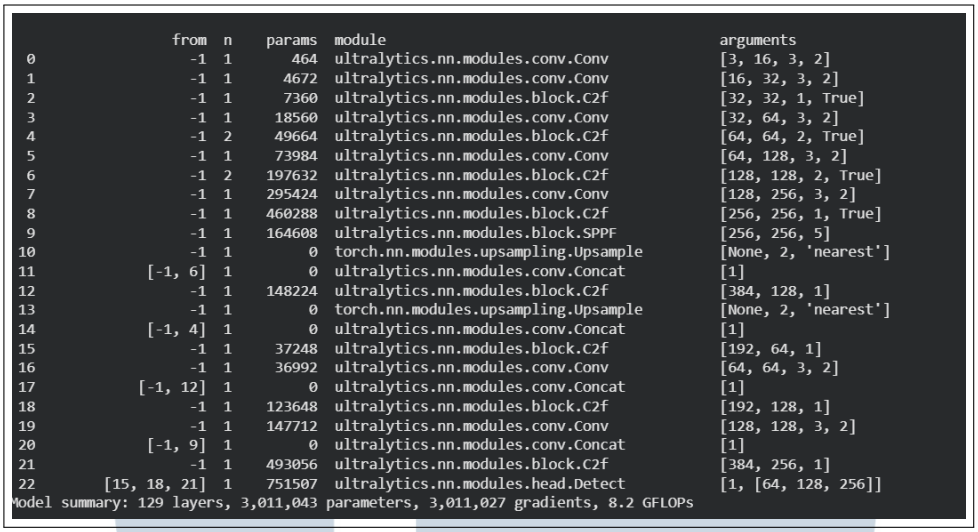
B *Synthetic training for detector & classifier*

Synthetic training disiapkan untuk mengatasi keterbatasan data hasil *scan* nyata, terutama pada tahap awal pengembangan. Pendekatan ini memungkinkan model belajar struktur dasar *bubble*, bentuk *ring*, dan pola isian tanpa harus mengumpulkan banyak dokumen asli. Prosesnya dimulai dengan membuat halaman sintetis yang meniru formulir inspeksi nyata, lengkap dengan dua blok tabel serta variasi visual seperti rotasi ringan, perbedaan tinta, dan *noise* pemindaian. Contoh hasil halaman sintetis ditunjukkan pada Gambar 3.8.



Gambar 3.8. Contoh halaman sintetis yang digunakan untuk melatih pendeteksi YOLO

Model *object detection* yang digunakan adalah YOLO. Sebelum proses *training* dimulai, YOLO menghasilkan *model summary* yang merangkum arsitektur lengkap detektor. Ringkasan ini ditunjukkan pada Gambar 3.9. Informasi yang ditampilkan mencakup susunan *layer*, jumlah parameter, modul utama seperti *Conv*, *C2f*, *SPPF*, serta kompleksitas komputasi yang dihitung sebagai GFLOPs. Ringkasan ini menunjukkan bahwa detektor yang dilatih merupakan model berukuran ringan namun cukup dalam untuk mempelajari pola visual *bubble* pada formulir inspeksi.



```
                 from  n    params  module                                       arguments
  0                -1  1       464  ultralytics.nn.modules.conv.Conv             [3, 16, 3, 2]
  1                -1  1      4672  ultralytics.nn.modules.conv.Conv             [16, 32, 3, 2]
  2                -1  1      7360  ultralytics.nn.modules.block.C2f             [32, 32, 1, True]
  3                -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  4                -1  2     49664  ultralytics.nn.modules.block.C2f             [64, 64, 2, True]
  5                -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  6                -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  7                -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  8                -1  1    460288  ultralytics.nn.modules.block.C2f             [256, 256, 1, True]
  9                -1  1    164608  ultralytics.nn.modules.block.SPPF            [256, 256, 5]
 10                -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11            [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 13                -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14            [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                -1  1     37248  ultralytics.nn.modules.block.C2f             [192, 64, 1]
 16                -1  1     36992  ultralytics.nn.modules.conv.Conv             [64, 64, 3, 2]
 17           [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                -1  1    123648  ultralytics.nn.modules.block.C2f             [192, 128, 1]
 19                -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 20            [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
 22      [15, 18, 21]  1    751507  ultralytics.nn.modules.head.Detect           [1, [64, 128, 256]]
Model summary: 129 layers, 3,011,043 parameters, 3,011,027 gradients, 8.2 GFLOPs
```

Gambar 3.9. Ringkasan arsitektur YOLO yang digunakan sebagai detektor pada tahap *synthetic training*

Selama *training*, YOLO menghasilkan *training logs* yang berisi perkembangan metrik pada setiap *epoch*. Contoh *log* tersebut ditunjukkan pada Gambar 3.10. Beberapa metrik utama yang muncul dalam *log* adalah:

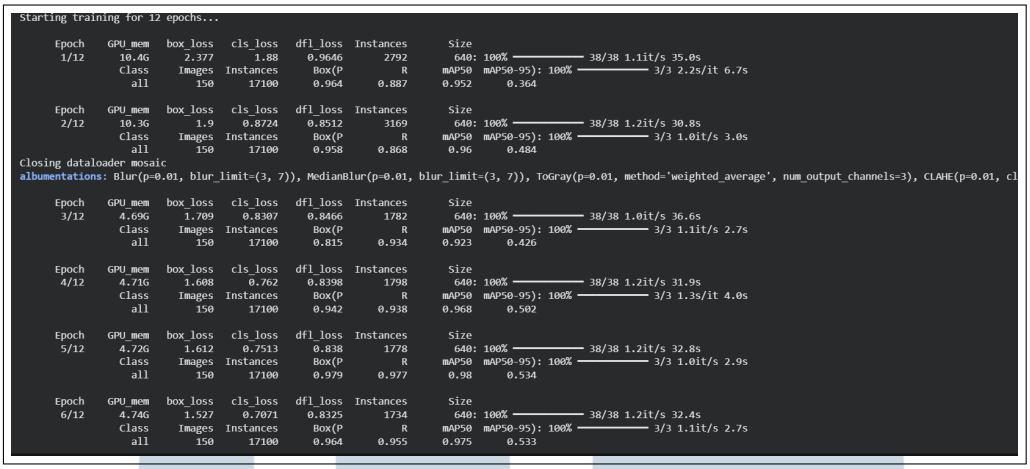
1. *box_loss*: error pada regresi koordinat *bounding box*. Nilai yang mengecil menunjukkan prediksi lokasi *bubble* semakin akurat.

2. *cls_loss*: error pada klasifikasi objek. Penurunan nilai ini menandakan model semakin tepat membedakan *bubble* dari *background*.

3. *dfl_loss*: *distribution focal loss* yang digunakan YOLO untuk mempelajari distribusi jarak guna meningkatkan ketelitian prediksi koordinat.

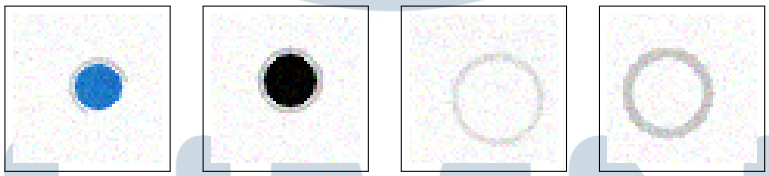
4. *mAP50* dan *mAP50–95*: metrik akurasi yang mengevaluasi kualitas deteksi berdasarkan IoU. Nilai yang meningkat menunjukkan model semakin bagus.

5. *Instances* dan *Images*: jumlah sampel yang diproses pada setiap *batch*.



Gambar 3.10. *Log training* YOLO pada *dataset* sintetis. *Log* menampilkan perkembangan *loss* dan mAP pada setiap epoch

Selain pendeteksi, *pipeline* membutuhkan *patch-level CNN* untuk mengklasifikasi apakah sebuah *bubble* terisi atau kosong. Untuk kebutuhan ini dibuat ratusan ribu *crop* berukuran 48×48 piksel dengan berbagai variasi isian seperti titik kecil, garis tipis, tanda silang, serta isian penuh. Contoh *crop* sintetis ditampilkan pada Gambar 3.11.



Gambar 3.11. Contoh *crop* sintetis untuk melatih *patch-level classifier*

Model *patch-level CNN* juga menjalani *training* awal menggunakan *dataset* sintetis tersebut. Contoh *training log* ditunjukkan pada Gambar 3.12. *Log* ini menampilkan perkembangan *train loss* dan *validation accuracy* pada setiap epoch. Nilai *train loss* yang terus menurun menunjukkan model berhasil mempelajari pola dasar isian, sementara peningkatan *validation accuracy* menandakan model tidak mengalami *overfit*.

```
2025-11-26 18:27:29,373 | INFO | PatchCNN 1/6 | train_loss 0.6730 | val_acc 0.729
INFO:omr_pro:PatchCNN 1/6 | train_loss 0.6730 | val_acc 0.729
2025-11-26 18:27:34,160 | INFO | PatchCNN 2/6 | train_loss 0.5177 | val_acc 0.866
INFO:omr_pro:PatchCNN 2/6 | train_loss 0.5177 | val_acc 0.866
2025-11-26 18:27:39,637 | INFO | PatchCNN 3/6 | train_loss 0.3746 | val_acc 0.908
INFO:omr_pro:PatchCNN 3/6 | train_loss 0.3746 | val_acc 0.908
2025-11-26 18:27:44,332 | INFO | PatchCNN 4/6 | train_loss 0.2673 | val_acc 0.953
INFO:omr_pro:PatchCNN 4/6 | train_loss 0.2673 | val_acc 0.953
2025-11-26 18:27:49,719 | INFO | PatchCNN 5/6 | train_loss 0.2000 | val_acc 0.956
INFO:omr_pro:PatchCNN 5/6 | train_loss 0.2000 | val_acc 0.956
2025-11-26 18:27:54,458 | INFO | PatchCNN 6/6 | train_loss 0.1542 | val_acc 0.977
INFO:omr_pro:PatchCNN 6/6 | train_loss 0.1542 | val_acc 0.977
2025-11-26 18:27:54,463 | INFO | PatchCNN best: /content/omr_pro/classifier/patch_cnn_best.pt
INFO:omr_pro:PatchCNN best: /content/omr_pro/classifier/patch_cnn_best.pt
```

Gambar 3.12. *Log training* TinyPatchCNN pada dataset sintetis. *Log* menunjukkan perkembangan *loss* dan akurasi di setiap *epoch*

Setelah tahap *training* sintetis selesai, model YOLO dan *classifier* telah mempelajari pola dasar *bubble* dengan cukup baik. Tahap ini menjadi fondasi sebelum dilanjutkan ke proses *fine-tuning* menggunakan *real scanned crops* agar model dapat beradaptasi dengan kondisi lapangan seperti tinta tidak merata, *noise* sensor, dan variasi gaya penandaan operator.

C *Real crop fine-tuning*

Tahap ini disiapkan untuk menyesuaikan *classifier* dengan pola isian *bubble* yang benar-benar muncul pada hasil *scan* lapangan. Meskipun model awal telah dilatih menggunakan *synthetic dataset*, karakteristik tinta, *noise* sensor, dan gaya coretan operator sering kali berbeda dari data sintetis. Karena itu, sistem membutuhkan *fine-tuning* menggunakan *real scanned crops* agar model lebih akurat dalam mengenali pola isian aktual. Contoh beberapa *crop* asli yang digunakan pada tahap ini ditunjukkan pada Gambar 3.13.

Gambar 3.13. Contoh *real scanned crops* untuk proses *fine-tuning* TinyPatchCNN

Proses *fine-tuning* dimulai dengan mengumpulkan tiga kelompok *crop* asli, yaitu *empty*, *filled*, dan *negative*. Kelas *negative* berisi potongan kecil halaman yang tidak mengandung *bubble*, misalnya garis tabel atau area kosong yang ikut ter-*crop*

pada saat deteksi. Pada tahap ini, kelas *negative* dipetakan ulang menjadi *empty* karena keduanya merepresentasikan kondisi tidak terisi.

Setelah data asli terkumpul, *dataset* dibagi menjadi *train* dan *validation*. Sebagian *train set* diberikan augmentasi ringan seperti rotasi kecil, perubahan warna di ruang HSV, blur tipis, atau penambahan *noise*. Teknik augmentasi ini membuat model lebih tahan terhadap variasi kualitas *scan* dan perbedaan pengisian di lapangan. Semua potongan gambar kemudian diubah ukurannya menjadi 48×48 piksel agar sesuai dengan arsitektur TinyPatchCNN.

Model *classifier* hasil *training* sintetis digunakan sebagai titik awal, lalu bobotnya diperbarui menggunakan *learning rate* kecil agar model tidak kehilangan pola dasar yang sudah dipelajari. Proses pembaruan ini berlangsung sambil memantau akurasi pada *validation set* dan menghentikan *training* lebih awal jika akurasi tidak lagi meningkat. Pendekatan ini memastikan bahwa model tidak mengalami *overfitting* pada sampel lapangan yang jumlahnya lebih sedikit.

Hasil akhir dari tahap ini adalah *classifier* yang jauh lebih stabil ketika memproses data operasi nyata. Model mampu membedakan berbagai pola isian dengan lebih akurat, mulai dari titik kecil yang samar, tanda silang, coretan tidak beraturan, hingga *bubble* kosong yang memiliki *ring* sangat tipis. Peningkatan akurasi ini berdampak langsung pada kualitas prediksi di tahap *inference*, terutama ketika sistem harus menetapkan keputusan akhir untuk setiap baris pada formulir inspeksi.

D *Inference*

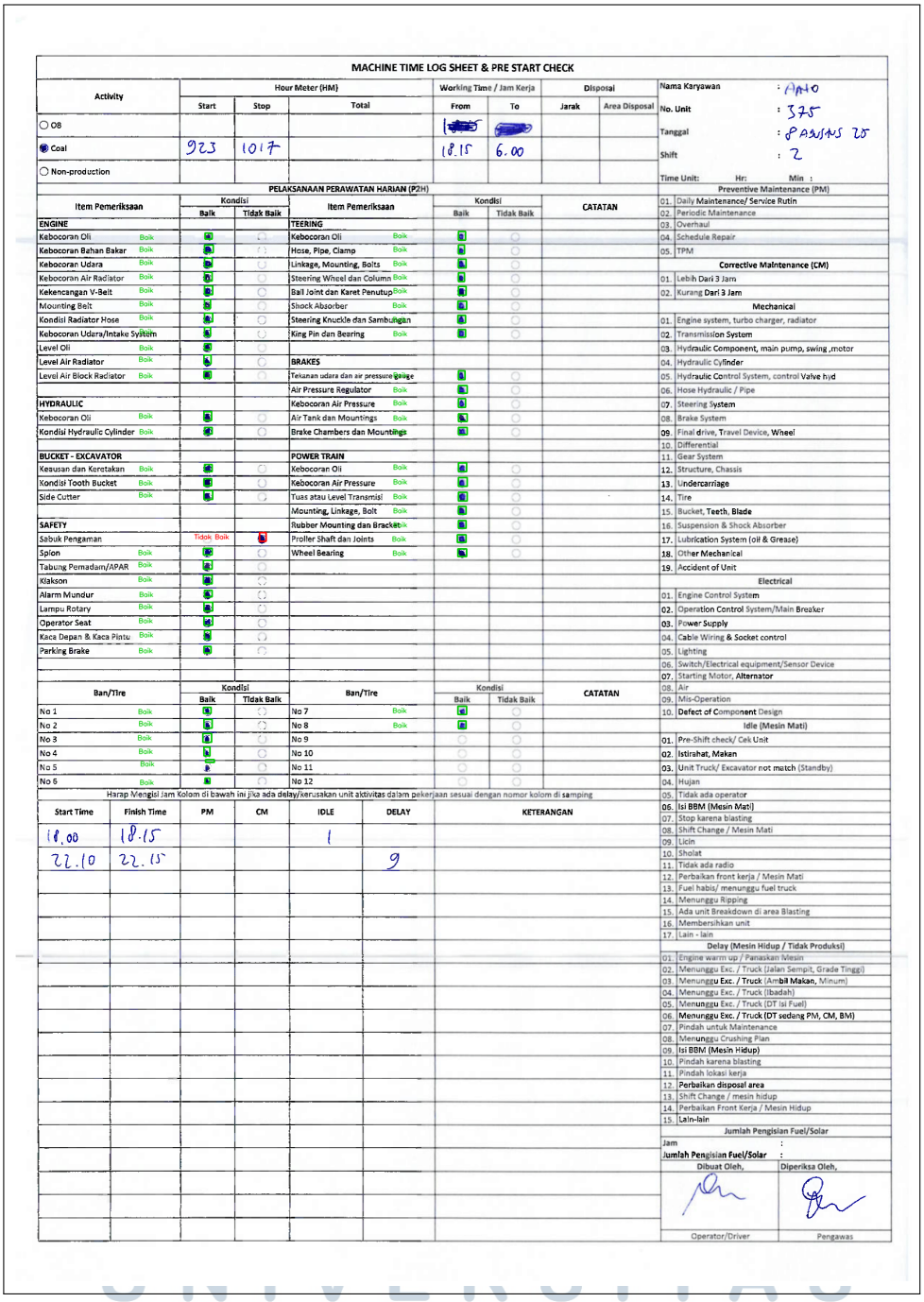
Tahap *inference* dimulai dengan membaca setiap *file* hasil *upload* dan mengonversinya menjadi citra BGR beresolusi tinggi. Sistem kemudian memetakan posisi 57 baris pertanyaan menggunakan *layout* yang telah disiapkan sebelumnya. Dari setiap posisi tersebut, diambil potongan kecil citra yang hanya berisi dua *bubble* pada baris tersebut. Potongan inilah yang kemudian dikirim ke model pendeteksi YOLO sehingga proses deteksi berlangsung pada area yang benar-benar relevan dan tidak terganggu oleh elemen lain di halaman.

Setelah YOLO menghasilkan beberapa kandidat *bubble*, sistem memilih satu kandidat terbaik untuk sisi kiri dan satu kandidat terbaik untuk sisi kanan. Pemilihan ini mempertimbangkan dua aspek sekaligus, yaitu skor deteksi dan jarak kandidat terhadap titik tengah kolom *bubble* yang ditentukan dalam *layout*.

Kedua kandidat dari setiap baris kemudian diklasifikasikan menggunakan

patch-level CNN. Sistem memotong area sekitar *bubble*, mengubahnya ke ukuran tetap, lalu meminta model menilai apakah *bubble* tersebut terisi atau kosong. Model menghasilkan probabilitas *filled*, sehingga keputusan tidak hanya didasarkan pada keberadaan tinta, tetapi juga tingkat keyakinan. Pendekatan dua tahap ini membuat sistem mampu menangani pola isian yang beragam, seperti titik kecil, tanda silang, atau coretan sangat tipis.

Tahap terakhir adalah penggabungan hasil sisi kiri dan kanan untuk menetapkan label baris. Jika sisi kiri terisi dan sisi kanan kosong, baris diberi label Baik. Jika sisi kanan terisi dan sisi kiri kosong, hasilnya menjadi Tidak Baik. Jika keduanya kosong, baris dinilai Kosong. Pada kasus langka ketika kedua sisi tampak terisi sekaligus, sistem memilih sisi dengan probabilitas tertinggi agar keputusan lebih konsisten. Contoh visualisasi hasil *inference* yang menunjukkan deteksi dan pelabelan akhir ditampilkan pada Gambar 3.14.



MACHINE TIME LOG SHEET & PRE START CHECK

Activity	Hour Meter (HM) Start	Stop	Total	Working Time / Jam Kerja From	To	Disposal Jarak	Area Disposal
○ OB							
◉ Coal	923	1017		18.15	6.00		
○ Non-production							

Nama Karyawan : [illegible]
No. Unit : 375
Tanggal : 8 AGUSTUS 25
Shift : 2
Time Unit: Hr: Min :

PELAKSANAAN PERAWATAN HARIAN (P2H)

Item Pemeriksaan	Kondisi Baik	Tidak Baik	Item Pemeriksaan	Kondisi Baik	Tidak Baik	CATATAN
ENGINE			**TEERING**			
Kebocoran Oli	Baik		Kebocoran Oli	Baik		
Kebocoran Bahan Bakar	Baik		Hose, Pipe, Clamp	Baik		
Kebocoran Udara	Baik		Linkage, Mounting, Bolts	Baik		
Kebocoran Air Radiator	Baik		Steering Wheel dan Column	Baik		
Kekencangan V-Belt	Baik		Ball Joint dan Karet Penutup	Baik		
Mounting Belt	Baik		Shock Absorber	Baik		
Kondisi Radiator Hose	Baik		Steering Knuckle dan Sambungan			
Kebocoran Udara/Intake System	Baik		King Pin dan Bearing	Baik		
Level Oli	Baik					
Level Air Radiator	Baik		**BRAKES**			
Level Air Block Radiator	Baik		Tekanan udara dan air pressure gauge			
			Air Pressure Regulator	Baik		
HYDRAULIC			Kebocoran Air Pressure	Baik		
Kebocoran Oli	Baik		Air Tank dan Mountings	Baik		
Kondisi Hydraulic Cylinder	Baik		Brake Chambers dan Mountings			
BUCKET - EXCAVATOR			**POWER TRAIN**			
Keausan dan Keretakan	Baik		Kebocoran Oli	Baik		
Kondisi Tooth Bucket	Baik		Kebocoran Air Pressure	Baik		
Side Cutter	Baik		Tuas atau Level Transmisi	Baik		
			Mounting, Linkage, Bolt	Baik		
SAFETY			Rubber Mounting dan Bracket	Baik		
Sabuk Pengaman	Tidak Baik		Proller Shaft dan Joints	Baik		
Spion	Baik		Wheel Bearing	Baik		
Tabung Pemadam/APAR	Baik					
Klakson	Baik					
Alarm Mundur	Baik					
Lampu Rotary	Baik					
Operator Seat	Baik					
Kaca Depan & Kaca Pintu	Baik					
Parking Brake	Baik					

Ban/Tire	Kondisi Baik	Tidak Baik	Ban/Tire	Kondisi Baik	Tidak Baik	CATATAN
No 1	Baik		No 7	Baik		
No 2	Baik		No 8	Baik		
No 3	Baik		No 9			
No 4	Baik		No 10			
No 5	Baik		No 11			
No 6	Baik		No 12			

Harap Mengisi Jam Kolom di bawah ini jika ada delay/kerusakan unit aktivitas dalam pekerjaan sesuai dengan nomor kolom di samping

Start Time	Finish Time	PM	CM	IDLE	DELAY	KETERANGAN
18.00	18.15			1		
22.10	22.15				9	

Preventive Maintenance (PM)
01. Daily Maintenance/ Service Rutin
02. Periodic Maintenance
03. Overhaul
04. Schedule Repair
05. TPM

Corrective Maintenance (CM)
01. Lebih Dari 3 Jam
02. Kurang Dari 3 Jam

Mechanical
01. Engine system, turbo charger, radiator
02. Transmission System
03. Hydraulic Component, main pump, swing ,motor
04. Hydraulic Cylinder
05. Hydraulic Control System, control Valve hyd
06. Hose Hydraulic / Pipe
07. Steering System
08. Brake System
09. Final drive, Travel Device, Wheel
10. Differential
11. Gear System
12. Structure, Chassis
13. Undercarriage
14. Tire
15. Bucket, Teeth, Blade
16. Suspension & Shock Absorber
17. Lubrication System (oil & Grease)
18. Other Mechanical
19. Accident of Unit

Electrical
01. Engine Control System
02. Operation Control System/Main Breaker
03. Power Supply
04. Cable Wiring & Socket control
05. Lighting
06. Switch/Electrical equipment/Sensor Device
07. Starting Motor, Alternator
08. Air
09. Mis-Operation
10. Defect of Component Design

Idle (Mesin Mati)
01. Pre-Shift check/ Cek Unit
02. Istirahat, Makan
03. Unit Truck/ Excavator not match (Standby)
04. Hujan
05. Tidak ada operator
06. Isi BBM (Mesin Mati)
07. Stop karena blasting
08. Shift Change / Mesin Mati
09. Licin
10. Sholat
11. Tidak ada radio
12. Perbaikan front kerja / Mesin Mati
13. Fuel habis/ menunggu fuel truck
14. Menunggu Ripping
15. Ada unit Breakdown di area Blasting
16. Membersihkan unit
17. Lain - lain

Delay (Mesin Hidup / Tidak Produksi)
01. Engine warm up / Panaskan Mesin
02. Menunggu Exc. / Truck (Jalan Sempit, Grade Tinggi)
03. Menunggu Exc. / Truck (Ambil Makan, Minum)
04. Menunggu Exc. / Truck (Ibadah)
05. Menunggu Exc. / Truck (DT Isi Fuel)
06. Menunggu Exc. / Truck (DT sedang PM, CM, BM)
07. Pindah untuk Maintenance
08. Menunggu Crushing Plan
09. Isi BBM (Mesin Hidup)
10. Pindah karena blasting
11. Pindah lokasi kerja
12. Perbaikan disposal area
13. Shift Change / mesin hidup
14. Perbaikan Front Kerja / Mesin Hidup
15. Lain-lain

Jumlah Pengisian Fuel/Solar
Jam :
Jumlah Pengisian Fuel/Solar :

Dibuat Oleh,	Diperiksa Oleh,
Operator/Driver	Pengawas

Gambar 3.14. Contoh hasil *inference* yang menampilkan bounding box berwarna sesuai label *Baik*, *Tidak Baik*, dan *Kosong*

E *Output to CSV*

Tahap akhir dari *pipeline* berfungsi untuk menyusun seluruh hasil prediksi menjadi format yang mudah dianalisis. Setelah model pendeteksi dan *classifier* menghasilkan keputusan untuk setiap baris, sistem menggabungkan informasi

tersebut ke dalam sebuah tabel yang berisi nama *file*, pertanyaan, label yang dipilih, tingkat keyakinan, serta koordinat *bounding box*. Struktur tabel ini kemudian diekspor menjadi *file* CSV agar dapat digunakan untuk inspeksi lanjutan atau integrasi ke *workflow* operasional. Contoh hasil *output* CSV ditunjukkan pada Gambar 3.15.

	file_name	question	selected option	confidence	bbox
0	Document_250821_143707-1.png	Kebocoran Oli	Baik	1.0000	355,405,13,11
1	Document_250821_143707-1.png	Kebocoran Bahan Bakar	Baik	1.0000	354,427,16,14
2	Document_250821_143707-1.png	Kebocoran Udara	Baik	1.0000	354,451,14,16
3	Document_250821_143707-1.png	Kebocoran Air Radiator	Baik	1.0000	354,477,14,16
4	Document_250821_143707-1.png	Kekencangan V-Belt	Baik	1.0000	354,502,15,16
5	Document_250821_143707-1.png	Mounting Belt	Baik	1.0000	354,527,11,16
6	Document_250821_143707-1.png	Kondisi Radiator Hose	Baik	1.0000	354,552,15,16
7	Document_250821_143707-1.png	Kebocoran Udara/Intake System	Baik	1.0000	354,578,11,16
8	Document_250821_143707-1.png	Level Oli	Baik	1.0000	354,608,12,11
9	Document_250821_143707-1.png	Level Air Radiator	Baik	1.0000	353,629,13,16
10	Document_250821_143707-1.png	Level Air Block Radiator	Baik	1.0000	353,659,13,12
11	Document_250821_143707-1.png	Kebocoran Oli	Baik	1.0000	353,733,14,16
12	Document_250821_143707-1.png	Kondisi Hydraulic Cylinder	Baik	1.0000	354,762,14,12
13	Document_250821_143707-1.png	Keausan dan Keretakan	Baik	1.0000	354,830,14,13
14	Document_250821_143707-1.png	Kondisi Tooth Bucket	Baik	1.0000	353,856,14,13
15	Document_250821_143707-1.png	Side Cutter	Baik	1.0000	353,878,16,16
16	Document_250821_143707-1.png	Sabuk Pengaman	Tidak Baik	1.0000	455,955,12,16

Gambar 3.15. Contoh tabel hasil *inference* yang diekspor ke format CSV

3.5.4 Evaluasi dan Pengajuan Hasil Model

Evaluasi sistem OMR pada tahap ini dilakukan melalui inspeksi visual terhadap hasil *inference*, dengan membandingkan posisi *bounding box*, warna penanda label, serta hasil klasifikasi terhadap kondisi formulir hasil *scan*. Pendekatan ini memastikan bahwa deteksi *bubble* dan penentuan label per baris telah sesuai secara visual, serta konsisten antara visualisasi hasil dan *output* CSV yang dihasilkan.

Hasil evaluasi tersebut kemudian diajukan kepada *supervisor* melalui sesi demonstrasi *pipeline* OMR menggunakan formulir uji yang merepresentasikan variasi kualitas *scan* dan pola isian operator. Pada proses pengajuan, *supervisor* meninjau kesesuaian hasil deteksi *bubble* pada setiap baris, ketepatan label akhir (Baik, Tidak Baik, atau Kosong), serta keterbacaan visualisasi pada citra asli. Selain itu, *output* CSV turut diperiksa untuk memastikan struktur kolom, konsistensi isi per baris, dan kesesuaian hasil dengan visualisasi. *Feedback* yang diperoleh digunakan

untuk melakukan penyesuaian *guardrails*, *threshold* klasifikasi, serta perbaikan antarmuka agar *pipeline* lebih stabil pada kondisi nyata. Setelah iterasi perbaikan, sistem dinilai telah memenuhi kebutuhan pada tahap *proof of concept* dan siap untuk dilanjutkan ke tahap integrasi serta evaluasi lanjutan.

3.6 Kendala dan Solusi yang Ditemukan

Selama pengembangan sistem RAG dan OMR, terdapat beberapa kendala utama yang memengaruhi proses implementasi, yaitu:

1. Pada pengembangan RAG, keterbatasan GPU dan memori pada lingkungan *development* membatasi pilihan model LLM dan *embedding* yang dapat dijalankan.
2. Pada pengembangan OMR, jumlah data *scan* nyata yang terbatas menyebabkan model deteksi dan klasifikasi belum dapat langsung mempelajari pola isian operator di lapangan.

Untuk mengatasi kendala tersebut, langkah-langkah berikut diterapkan agar proses pengembangan tetap optimal dan hasil sistem lebih konsisten.

1. Untuk RAG, dipilih model LLM dan *embedding* yang lebih efisien namun tetap kompetitif berdasarkan hasil *benchmark*, sehingga dapat dijalankan dengan keterbatasan sumber daya komputasi.
2. Untuk OMR, dikembangkan *synthetic dataset* sebagai dasar pelatihan model, kemudian dilakukan *fine-tuning* menggunakan *real scanned crops*. Pendekatan ini membantu model beradaptasi dengan variasi tinta, *noise*, dan pola isian aktual meskipun data nyata masih terbatas.