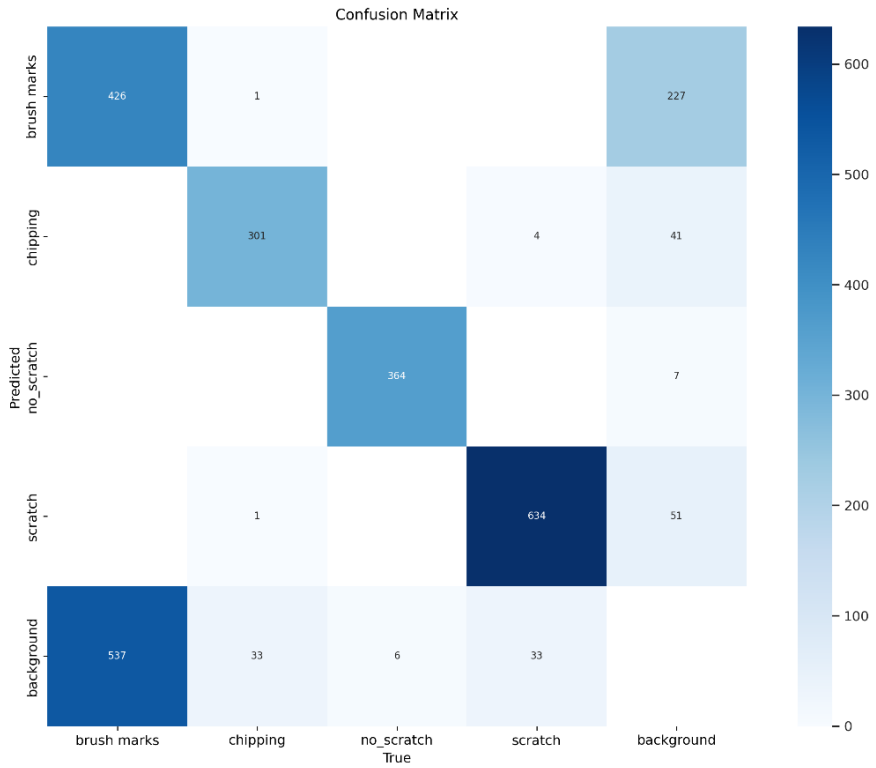
BAB IV
PENGUJIAN DAN ANALISA

4.1.Analisa Hasil Training YoloV8

Setelah melalui tahap *training* pada sub bab 3.6.2. (BAB III), di dapatkan hasil *training* untuk kedua model YOLOv8. Hasil tersebut diantaranya *confusion matrix*, *f1 curve*, *labels*, *labels correlogram*, *precision curve, , recall curve, precision recall curve,* dan kurva training.

4.1.1. *Confusion Matrix*

Confusion Matrix adalah alat evaluasi yang penting untuk mendapatkan gambaran komprehensif mengenai seberapa baik model dapat memprediksi kelas-kelas yang berbeda Pada penelitian ini dihasilkan *confusion matrix* sesuai pada **Gambar 4.1** dan **Gambar 4.2**.



Gambar 4.1 *Confusion Matrix* Pelatihan Pertama

Gambar 4.2 *Confusion Matrix* Pelatihan Kedua

Confusion matrix menghasilkan ringkasan visual dari hasil prediksi. Kedua *confusion matrix* diatas menggunakan jumlah dataset yang sama Dari hasil tersebut diperoleh perhitungan sebagai berikut:

a. Pelatihan Pertama

Berdasarkan **Gambar 4.1** diperoleh perhitungan akurasi, presisi, recall, dan *F1-score* sebagai berikut,

1) Akurasi (*Accuracy*)

Dengan menggunakan persamaan (2.4) diperoleh akurasi sebagai berikut,

$$Akurasi_1 = \frac{\sum Matriks\ True}{N}$$

$$= \frac{TP_{brush\ marks1}+TP_{chipping1}+TP_{no\_scratch1}+TP_{scratch1}+TP_{background1}}{N}$$

$$= \frac{426+301+364+634+51}{426+1+227+301+41+364+7+1+634+51+537+33+6+33+51}$$

$$Akurasi_1 = \frac{1776}{2397} \approx 0{,}7409$$

2) Presisi (*Precision*)

Dengan menggunakan persamaan (2.5) diperoleh presisi per kelas sebagai berikut,

- *Brush marks*

$$Presisi_{brush\ marks1} = \frac{\mathrm{TP}_{brush\ marks1}}{\mathrm{TP}_{brush\ marks1}+FP_{brush\ marks1}}$$

$$Presisi_{brush\ marks1} = \frac{426}{426+1+227} = \frac{426}{654} \approx 0{,}6514$$

- *Chipping*

$$Presisi_{chipping1} = \frac{TP_{chipping1}}{TP_{chipping1}+FP_{chipping1}}$$

$$Presisi_{chipping1} = \frac{301}{301+1+4+41} = \frac{301}{347} \approx 0{,}8674$$

- *No_scratch*

$$Presisi_{no\_scratch1} = \frac{TP_{no\_scratch1}}{TP_{no\_scratch1}+FP_{no\_scratch1}}$$

$$Presisi_{no\_scratch1} = \frac{364}{364+7} = \frac{364}{371} \approx 0{,}9811$$

- *Scratch*

$$Presisi_{scratch1} = \frac{TP_{scratch1}}{TP_{scratch1}+FP_{scratch1}}$$

$$Presisi_{scratch1} = \frac{634}{634+1+51} = \frac{634}{686} \approx 0{,}9242$$

- *Background*

$$Presisi_{background1} = \frac{TP_{background1}}{TP_{background1}+FP_{background1}}$$

$$Presisi_{background1} = \frac{51}{51+537+33+6+33} = \frac{51}{660} \approx 0{,}0773$$

3) *Recall*

Dengan menggunakan persamaan (2.6) diperoleh *recall* per kelas sebagai berikut,

- *Brush marks*

$$Recall_{brush\ marks1} = \frac{\mathrm{TP}_{brush\ marks1}}{\mathrm{TP}_{brush\ marks1}+FN_{brush\ marks1}}$$

$$Recall_{brush\ marks1} = \frac{426}{426+1+537} = \frac{426}{964} \approx 0{,}4419$$

- *Chipping*

$$Recall_{chipping1} = \frac{TP_{chipping1}}{TP_{chipping1}+FN_{chipping1}}$$

$$Recall_{chipping1} = \frac{301}{301+1+33} = \frac{301}{335} \approx 0{,}8985$$

- *No_scratch*

$$Recall_{no\_scratch1} = \frac{TP_{no\_scratch1}}{TP_{no\_scratch1}+FN_{no\_scratch1}}$$

$$Recall_{no\_scratch1} = \frac{364}{364+6} = \frac{364}{370} \approx 0{,}9838$$

- *Scratch*

$$Recall_{scratch1} = \frac{TP_{scratch1}}{TP_{scratch1}+FN_{scratch1}}$$

$$Recall_{scratch1} = \frac{634}{634+4+33} = \frac{634}{671} \approx 0{,}9449$$

- *Background*

$$Recall_{background1} = \frac{TP_{background1}}{TP_{background1}+FN_{background1}}$$

$$Recall_{background1} = \frac{51}{51+227+41+7+51} = \frac{51}{326} \approx 0{,}1564$$

4) *F1-Score*

Dengan menggunakan persamaan (2.7) diperoleh *F1-score* per kelas sebagai berikut,

- *Brush marks*

$$F1 - score_{brush\ marks1} = 2 \times \frac{Presisi_{brush\ marks1} \times Recall_{brush\ marks1}}{Presisi_{brush\ marks1}+Recall_{brush\ marks1}}$$

$$F1 - score_{brush\ marks1} = 2 \times \frac{0{,}6514 \times 0{,}4419}{0{,}6514+0{,}4419} \approx 0{,}5262$$

- *Chipping*

$$F1 - score_{chipping1} = 2 \times \frac{Presisi_{chipping1} \times Recall_{chipping1}}{Presisi_{chipping1}+Recall_{chipping1}}$$

$$F1 - score_{chipping1} = 2 \times \frac{0{,}8674 \times 0{,}8985}{0{,}8674+0{,}8985} \approx 0{,}8827$$

- *No_scratch*

$$F1 - score_{no\_scratch1} = 2 \times \frac{Presisi_{no\_scratch1} \times Recall_{no\_scratch1}}{Presisi_{no\_scratch1}+Recall_{no\_scratch1}}$$

$$F1 - score_{no\_scratch} = 2 \times \frac{0{,}9811 \times 0{,}9838}{0{,}9811+0{,}9838} \approx 0{,}9825$$

- *Scratch*

$$F1 - score_{scratch1} = 2 \times \frac{Presisi_{scratch1} \times Recall_{scratch1}}{Presisi_{scratch1}+Recall_{scratch1}}$$

$$F1 - score_{scratch1} = 2 \times \frac{0{,}9242 \times 0{,}9449}{0{,}9242+0{,}9449} \approx 0{,}9344$$

- *Background*

$$F1-score_{background1} = 2 \times \frac{Presisi_{background1} \times Recall_{background1}}{Presisi_{background1} + Recall_{background1}}$$

$$F1-score_{background1} = 2 \times \frac{0{,}0773 \times 0{,}1564}{0{,}0773 + 0{,}1564} \approx 0{,}1036$$

5) Rata-rata Matriks

Perhitungan matriks presisi, recall, dan F1-score dilakukan untuk setiap kelas secara terpisah, sehingga diperlukan perhitungan nilai rata-rata agar dapat menggambarkan secara umum performa keseluruhan dari masing-masing matriks tersebut. Berikut dibawah ini perhitungan rata-rata matriks,

$$Presisi_{rata-rata1} = \frac{(0{,}6514 + 0{,}8674 + 0{,}9811 + 0{,}9242 + 0{,}0773)}{5} \approx 0{,}6903$$

$$Recall_{rata-rata1} = \frac{(0{,}44191 + 0{,}8985 + 0{,}9838 + 0{,}9449 + 0{,}1564)}{5} \approx 0{,}6851$$

$$F1-score_{rata-rata1} = \frac{(0{,}5262 + 0{,}8827 + 0{,}9825 + 0{,}9344 + 0{,}1036)}{5} \approx 0{,}6859$$

i. Pelatihan Kedua

Berdasarkan gambar **Gambar 4.2** diperoleh perhitungan akurasi, presisi, *recall*, dan *F1-score* sebagai berikut,

1) Akurasi (*Accuracy*)

Dengan menggunakan persamaan (2.4) diperoleh perhitungan akurasi sebagai berikut,

$$Akurasi_2 = \frac{\sum Matriks\ True}{N}$$

$$= \frac{TP_{brush\ marks2} + TP_{chipping2} + TP_{no\_scratch2} + TP_{scratch2} + TP_{background2}}{N}$$

$$= \frac{458 + 307 + 361 + 632 + 35}{458 + 217 + 3 + 307 + 4 + 32 + 361 + 21 + 3 + 632 + 33 + 505 + 26 + 9 + 35}$$

$$Akurasi_2 = \frac{1793}{2651} \approx 0{,}6763$$

2) Presisi (*Precision*)

Dengan menggunakan persamaan (2.5) diperoleh perhitungan presisi per kelas sebagai berikut,

- *Brush marks*

$$Presisi_{brush\ marks2} = \frac{\mathrm{TP}_{brush\ marks2}}{\mathrm{TP}_{brush\ marks2} + FP_{brush\ marks2}}$$

$$Presisi_{brush\ marks2} = \frac{458}{458+217} = \frac{458}{675} \approx 0{,}6785$$

- *Chipping*

$$Presisi_{chipping2} = \frac{TP_{chipping2}}{TP_{chipping2}+FP_{chipping2}}$$

$$Presisi_{chipping2} = \frac{307}{307+4+32} = \frac{307}{343} \approx 0{,}8950$$

- *No_scratch*

$$Presisi_{no\_scratch2} = \frac{TP_{no\_scratch2}}{TP_{no\_scratch2}+FP_{no\_scratch2}}$$

$$Presisi_{no\_scratch2} = \frac{361}{361+21} = \frac{361}{382} \approx 0{,}9450$$

- *Scratch*

$$Presisi_{scratch2} = \frac{TP_{scratch2}}{TP_{scratch2}+FP_{scratch2}}$$

$$Presisi_{scratch2} = \frac{632}{632+3+33} = \frac{632}{668} \approx 0{,}9461$$

- *Background*

$$Presisi_{background2} = \frac{TP_{background2}}{TP_{background2}+FP_{background2}}$$

$$Presisi_{background2} = \frac{35}{35+505+26+9+35} = \frac{35}{575} \approx 0{,}0609$$

3) *Recall*

Dengan menggunakan persamaan (2.6) diperoleh perhitungan *recall* per kelas sebagai berikut,

- *Brush marks*

$$Recall_{brush\ marks2} = \frac{\mathrm{TP}_{brush\ marks2}}{\mathrm{TP}_{brush\ marks2}+FN_{brush\ marks2}}$$

$$Recall_{brush\ marks2} = \frac{458}{458+3+505} = \frac{458}{966} \approx 0{,}4741$$

- *Chipping*

$$Recall_{chipping2} = \frac{TP_{chipping2}}{TP_{chipping2}+FN_{chipping2}}$$

$$Recall_{chipping2} = \frac{307}{307+3+26} = \frac{307}{336} \approx 0{,}9137$$

- *No_scratch*

$$Recall_{no\_scratch2} = \frac{TP_{no\_scratch2}}{TP_{no\_scratch2}+FN_{no\_scratch2}}$$

$$Recall_{no\_scratch2} = \frac{361}{361+4+9} = \frac{361}{374} \approx 0{,}9652$$

- *Scratch*

$$Recall_{scratch2} = \frac{TP_{scratch2}}{TP_{scratch2}+FN_{scratch2}}$$

$$Recall_{scratch2} = \frac{632}{632+32+21+35} = \frac{632}{720} \approx 0{,}8778$$

- *Background*

$$Recall_{background2} = \frac{TP_{background2}}{TP_{background2}+FN_{background2}}$$

$$Recall_{background2} = \frac{35}{35+217+32+21+33} = \frac{35}{338} \approx 0{,}1036$$

4) *F1-Score*

Dengan menggunakan persamaan (2.7) diperoleh perhitungan *F1-score* per kelas sebagai berikut,

- *Brush marks*

$$F1 - score_{brush\ marks2} = 2 \times \frac{Presisi_{brush\ marks2} \times Recall_{brush\ marks2}}{Presisi_{brush\ marks2}+Recall_{brush\ marks2}}$$

$$F1 - score_{brush\ marks2} = 2 \times \frac{0{,}6785 \times 0{,}4741}{0{,}6785+0{,}4741} \approx 0{,}5583$$

- *Chipping*

$$F1 - score_{chipping2} = 2 \times \frac{Presisi_{chipping2} \times Recall_{chipping2}}{Presisi_{chipping2}+Recall_{chipping2}}$$

$$F1 - score_{chipping2} = 2 \times \frac{0{,}8950 \times 0{,}9137}{0{,}8950+0{,}9137} \approx 0{,}9043$$

- *No_scratch*

$$F1 - score_{no\_scratch2} = 2 \times \frac{Presisi_{no\_scratch2} \times Recall_{no\_scratch2}}{Presisi_{no\_scratch2}+Recall_{no\_scratch2}}$$

$$F1 - score_{no\_scratch2} = 2 \times \frac{0{,}9450 \times 0{,}9652}{0{,}9450+0{,}9652} \approx 0{,}9550$$

- *Scratch*

$$F1 - score_{scratch2} = 2 \times \frac{Presisi_{scratch2} \times Recall_{scratch2}}{Presisi_{scratch2}+Recall_{scratch2}}$$

$$F1 - score_{scratch2} = 2 \times \frac{0{,}9461 \times 0{,}8778}{0{,}9461+0{,}8778} \approx 0{,}9104$$

- *Background*

$$F1 - score_{background2} = 2 \times \frac{Presisi_{background2} \times Recall_{background2}}{Presisi_{background2}+Recall_{background2}}$$

$$F1 - score_{background2} = 2 \times \frac{0{,}0609 \times 0{,}1036}{0{,}0609+0{,}1036} \approx 0{,}0784$$

5) Rata-rata Matriks

Perhitungan matriks presisi, recall, dan F1-score dilakukan untuk setiap kelas secara terpisah, sehingga diperlukan perhitungan nilai rata-rata agar dapat menggambarkan secara umum performa keseluruhan dari masing-masing matriks tersebut. Berikut dibawah ini perhitungan rata-rata matriks,

$$Presisi_{rata-rata2} = \frac{(0{,}6785+0{,}8950+0{,}9450+0{,}9461+0{,}0609)}{5} \approx 0{,}7051$$

$$Recall_{rata-rata2} = \frac{(0{,}4741+0{,}9137+0{,}9652+0{,}8778+0{,}1036)}{5} \approx 0{,}6669$$

$$F1-score_{rata-rata2} = \frac{(0{,}5583+0{,}9043+0{,}9550+0{,}9107+0{,}0784)}{5} \approx 0{,}6813$$

Dari hasil perhitungan *confusion matrix* kedua pelatihan tersebut diperoleh ringkasan perhitungannya pada **Tabel 4.1**.

Tabel 4.1 Hasil Perhitungan *Confusion Matrix*

Pelatihan ke-	Akurasi	Presisi	Recall	F1-score
1	0,74	0,69	0,68	0,685
2	0,67	0,70	0,66	0,681

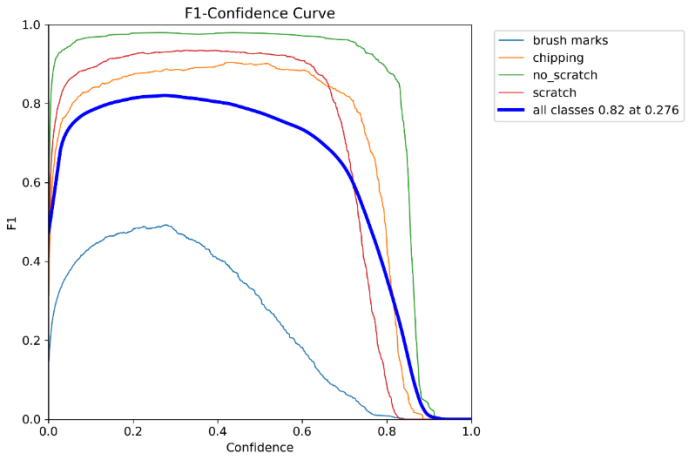
Perbandingan yang dihasilkan antara pelatihan model pertama dan kedua mengungkapkan adanya perubahan yang signifikan, di mana model kedua secara keseluruhan menunjukkan sedikit penurunan performa. Secara umum, terdapat penurunan akurasi yang cukup terasa, yaitu sebesar 6,46%, menjadikannya turun dari 74,09% menjadi 67,63%. Penurunan ini mencerminkan berkurangnya kemampuan model kedua untuk secara konsisten mengklasifikasikan semua sampel dengan benar.

Meskipun hasil rata-rata *F1-score* hampir tidak berubah, analisis per kelas menunjukkan adanya perubahan fokus model. Pada kelas *chipping* dan *brush marks* terdapat peningkatan. Namun, pada kelas *no_scratch* dan *scratch* terjadi penurunan. Selain itu, terjadi penurunan pada *background*, terutama *recall* dari *background* yang turun drastis dari 0,1564 menjadi 0,1036. Hal ini menyebabkan *F1-score* yang tadinya di dongkrak oleh kelas *chipping* dan *brush marks* menjadi turun. Akibat dari *background* yang turun ialah kesalahan interpretasi mana area

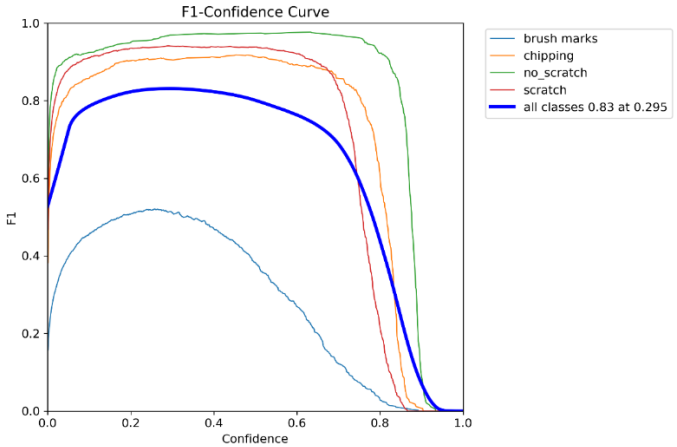
background dan mana yang area cacat. Hal ini menunjukkan bahwa pelatihan pertama menjadi pilihan yang lebih stabil daripada pelatihan kedua secara keseluruhan, terutama dari akurasi yang lebih besar 6,46%.

4.1.2. *F1-Confidence curve*

F1-Confidence curve yaitu grafik yang memvisualisasikan bagaimana nilai *F1-score* suatu model berubah seiring dengan perubahan *Confidence Threshold* dari 0,0 hingga 1,0. Tujuan dari grafik ini ialah untuk menemukan *Confidence Threshold* optimal yang menghasilkan *F1-score* tertinggi dan melihat seberapa stabil dan sensitif performa model terhadap perubahan *confidence threshold*. Grafik *F1-Confidence curve* pelatihan pertama dan kedua dapat dilihat pada **Gambar 4.3** dan **Gambar 4.4**.



Gambar 4.3 *F1 Curve* Pelatihan Pertama



Gambar 4.4 *F1 Curve* Pelatihan Kedua

Dari kedua grafik tersebut diperoleh bahwa model pada Pelatihan Kedua menunjukkan peningkatan potensi kinerja optimal dibandingkan Pelatihan Pertama, dengan kenaikan F1-score rata-rata keseluruhan dari 0,82 menjadi 0,83. Peningkatan kinerja puncak ini dicapai pada *Confidence Threshold* yang sedikit lebih tinggi, yakni 0,295 pada pelatihan kedua, berbanding 0,276 pada pelatihan pertama. Pergeseran ambang batas optimal ke nilai yang lebih tinggi mengindikasikan bahwa, untuk mencapai keseimbangan terbaik antara *precision* dan *recall*, model pada pelatihan kedua membutuhkan skor keyakinan minimal yang sedikit lebih tinggi dalam mengklasifikasikan hasil prediksinya.

Untuk menguji stabilitas model terhadap perubahan *threshold*, dilakukan perbandingan dengan *F1-score* yang diperoleh dari *Confusion Matrix* (CM) pada sub-bab 4.1.1. Perbedaan antara kedua nilai *F1-score* ini yakni *F1-score* maksimum (dari kurva) yang dicapai pada *threshold* optimal dan *F1-score* dari CM yang dicapai pada *threshold* tetap menggambarkan sensitivitas model. *F1-score* rata-rata dari CM pada pelatihan pertama adalah 0,6859. Berdasarkan proyeksi pada kurva rata-rata semua kelas, nilai *F1-score* tersebut sesuai dengan *Confidence Threshold* yang relatif tinggi, yaitu sekitar 0,75. Dengan demikian, model mengalami penurunan *F1-score* yang signifikan, yaitu dari 0,82 menjadi 0,6859, ketika *Confidence Threshold* dinaikkan dari 0,276 ke 0,75. Fenomena serupa terjadi pada pelatihan kedua, di mana kinerja F1-score rata-rata juga menurun pada *threshold* tinggi. Hal ini menunjukkan bahwa kedua model sangat sensitif terhadap *Confidence Threshold* yang tinggi, dan F1-score keduanya tidak stabil ketika ambang batas keyakinan diatur secara ketat di atas 0,70. Untuk mempermudah memahami dapat dilihat tabel analisa pada **Tabel 4.2**.

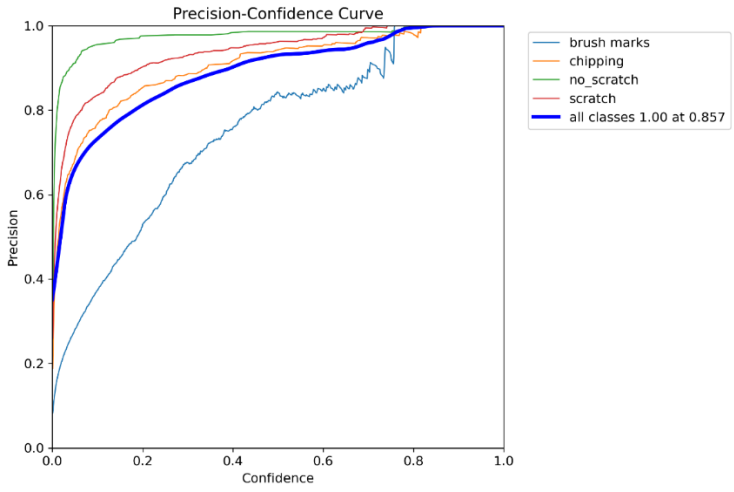
Tabel 4.2 Analisa F1-Confidence Curve dengan F1-score Confusion Matrix

Metrik Kinerja	**Pelatihan Pertama**	**Pelatihan Kedua**	**Keterangan**
F1-Curve	0,82	0,83	Performa terbaik yang dicapai model.

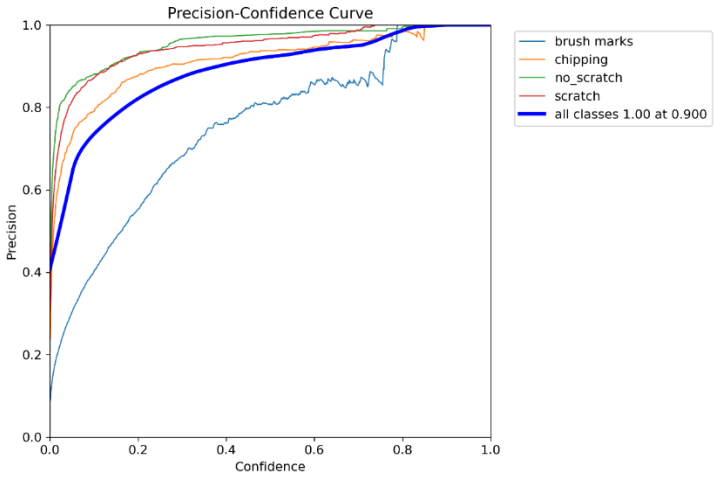
Metrik Kinerja	Pelatihan Pertama	Pelatihan Kedua	Keterangan
Confidence Threshold Optimal	0,276	0,295	*Confidence Threshold* yang digunakan untuk mencapai performa terbaik.
F1-Confusion Matrix	0,6859	0,6813	Nilai *F1* ketika menggunakan *confidence threshold* tetap yakni 0,75.
Penurunan F1-score (Kurva ke *Confusion Matrix*)	0,1341	0,1487	Penurunan F1-score pada pelatihan pertama dan kedua akibat *confidence threshold yang tinggi* (0,75).

4.1.3. *Precision-Confidence curve*

Precision-Confidence curve adalah kurva yang memvisualisasikan hubungan antara nilai presisi dengan *confidence threshold*. Dengan tujuan untuk mengetahui bagaimana caranya untuk mencapai presisi 1.00 dengan mengubah nilai *confidence threshold.* Untuk itu dapat dilihat kedua kurva hasil pelatihan pertama dan kedua pada **Gambar 4.5** dan **Gambar 4.6**.



Gambar 4.5 *Precision-Confidence Curve* Pelatihan Pertama



Gambar 4.6 *Precision-Confidence Curve* Pelatihan Kedua

Dari kedua kurva *Precision-Confidence Curve* tersebut dihasilkan hubungan yang berbanding terbalik secara umum antara presisi dan *Confidence Threshold*. Ketika *Confidence Threshold* ditingkatkan (bergerak ke kanan pada sumbu X), model hanya menerima prediksi yang memiliki keyakinan tinggi, yang secara otomatis mengurangi jumlah *False Positives* dan dengan demikian meningkatkan presisi. Sebaliknya, ketika *Confidence Threshold* diturunkan (bergerak ke kiri), lebih banyak prediksi, termasuk yang salah (*False Positives*), menyebabkan presisi menurun. Perbandingan pelatihan pertama dan kedua menunjukkan adanya peningkatan kinerja pada pelatihan kedua, karena kurva tebal biru ('all classes') pada pelatihan kedua berada sedikit di atas pelatihan pertama dan mencapai presisi 1.00 pada *Confidence Threshold* 0.900, dibandingkan 0.857 pada pelatihan pertama. Peningkatan ini menunjukkan bahwa model pada pelatihan kedua mampu mencapai Presisi tinggi dengan keyakinan yang lebih kuat (*confidence threshold* yang lebih tinggi), meskipun kelas *brush marks* tetap menjadi kelas dengan kinerja terburuk di kedua model.

Presisi yang terlihat pada kurva dengan rata-rata presisi dari confusion matrix pembahasan sub bab 4.1.1 (0.6903 dan 0.7051) adalah bahwa nilai rata-rata presisi hasil pengukuran kinerja model pada satu ambang batas keyakinan (*Confidence Threshold*) tunggal yang digunakan untuk menghasilkan *confusion matrix*. Berdasarkan pengamatan visual pada kurva: rata-rata presisi sebesar ≈ 0,70 (baik 0,6903 maupun 0,7051) terletak pada *Confidence Threshold* yang sangat

rendah, yaitu di kisaran 0,0 hingga 0,1 pada sumbu X di kedua kurva. Presisi yang dihasilkan pada ambang batas operasional CM (0.6903 dan 0.7051) tercapai hanya pada *Confidence Threshold* yang hampir nol, ini menggambarkan bahwa presisi aktual model sangat sensitif terhadap perubahan *threshold*. Jika model dioperasikan pada threshold yang lebih tinggi (misalnya 0,5, di mana presisi kurva berada di sekitar 0,88), presisi dari confusion matrix seharusnya akan jauh lebih tinggi dari 0.7051. Fenomena ini menunjukkan bahwa threshold yang digunakan untuk menghasilkan rata-rata presisi dalam confusion matrix adalah *confidence threshold* yang sangat rendah dan model akan mengalami peningkatan presisi yang signifikan jika ambang batas keyakinan diatur lebih tinggi. Untuk dapat melihat hasil analisanya dengan format lain dapat dilihat pada **Tabel 4.3**.

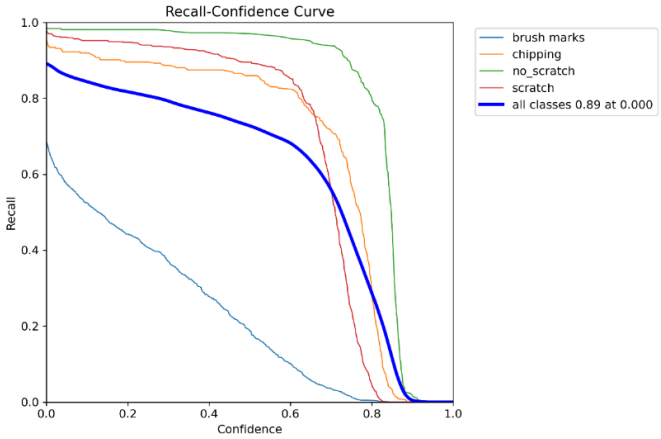
Tabel 4.3 Analisa Precision-Confidence Curve dengan Presisi Confusion Matrix

Metrik Kinerja	**Pelatihan Pertama**	**Pelatihan Kedua**	**Keterangan**
Precision-Confidence Curve	Mencapai 1,00	Mencapai 1,00	Presisi tertinggi yang dapat dicapai model
Confidence Threshold Presisi 1,00	≈ 0,857	≈0,900	Confidence threshold yang digunakan mengalami peningkatan di pelatihan kedua
Rata-rata Presisi Confusion Matrix	0,6903	0,7051	Nilai rata-rata presisi ketika *confidence threshold* berada di kisaran 0,0-0,1.
Peningkatan Presisi	≈ 0,30	≈ 0,30	Peningkatan presisi akibat perpindahan dari *confidence threshold* rendah ke tinggi.

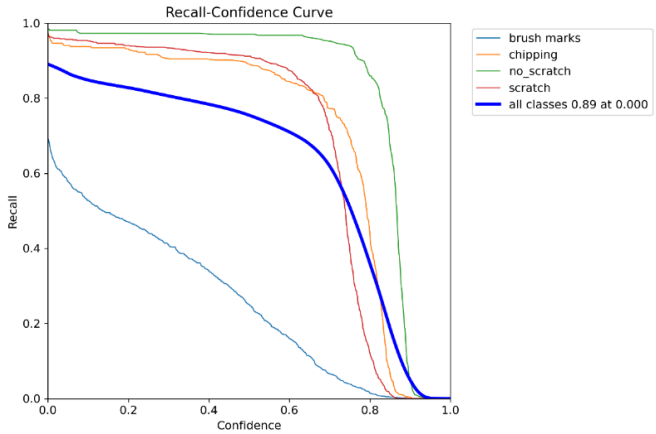
4.1.4. *Recall-Confidence curve*

Recall-Confidence curve adalah kurva yang memvisualisasikan hubungan antara *recall* dengan *confidence threshold*. Dengan tujuan untuk mengetahui pada *confidence threshold* berapakah yang paling sesuai untuk suatu model deteksi.

Kurva *recall-confidence curve* pelatihan pertama dan kedua dapat dilihat **Gambar 4.7** dan **Gambar 4.8**.



Gambar 4.7 Recall-Confidence Curve Pelatihan Pertama



Gambar 4.8 *Recall-Confidence Curve* Pelatihan Kedua

Kedua kurva *Recall-Confidence* dari pelatihan pertama dan pelatihan kedua menunjukkan hasil yang identik atau sangat mirip, menggambarkan hubungan antara *recall* (sensitivitas) dan *Confidence Threshold* untuk setiap kelas. Hubungan ini berbanding terbalik yaitu semakin tinggi nilai *Confidence Threshold* yang ditetapkan (bergerak ke kanan pada sumbu X), semakin rendah nilai *Recall* yang dicapai (bergerak ke bawah pada sumbu Y). Hal ini membuktikan bahwa untuk mengklasifikasikan prediksi sebagai positif, model dituntut memiliki *confidence* yang lebih tinggi, yang konsekuensinya akan mengurangi *True Positives* (sebab banyak prediksi dengan *confidence threshold* yang rendah kini ditolak), sehingga

recall menurun. Secara spesifik, kelas *no_scratch* (hijau) menunjukkan robustnes tertinggi dengan *recall* yang bertahan stabil, sementara kelas *brush marks* (biru muda) adalah yang paling sensitif terhadap kenaikan *threshold*. Kurva rata-rata *all classes* (biru tebal) pada kedua pelatihan menunjukkan performa model yang tidak berubah secara signifikan.

Analisis hubungan antara kurva *recall-confidence* dengan data *recall* dari *confusion matrix* menunjukkan bahwa performa yang dicatat sangat bergantung pada *confidence threshold* yang dipilih. Nilai rata-rata Recall maksimum model pada kedua pelatihan adalah 0.89 pada *threshold* 0.000, yang merupakan potensi tertinggi yang dapat dicapai model. Mengutip dari sub bab 4.1.1., pada pelatihan pertama, diperoleh perhitungan *recall* dari *confusion matrix* sebesar 0.6851 yang dicapai pada *confidence threshold* sekitar 0.61. Hal ini menunjukkan penurunan *recall* sebesar 0.2049 (dari 0.89 ke 0.6851). Sementara itu, pada pelatihan kedua, mengutip dari sub bab 4.1.1., diperoleh perhitungan *recall* dari *confusion matrix* sebesar 0.6669 yang dicapai pada *confidence threshold* sekitar 0.72. Penurunan *recall* yang terjadi pada pelatihan kedua sebesar 0.2231 (0.89 ke 0.6669) lebih besar karena *confidence threshold* yang digunakan dinaikkan lebih signifikan ke 0.72. Penurunan *recall confusion matrix* secara nominal dari 0.6851 menjadi 0.6669 bukanlah indikasi model memburuk, melainkan konsekuensi logis yang diharapkan dari kenaikan *confidence threshold*. Untuk dapat melihat hasil analisanya dengan format lain dapat dilihat pada **Tabel 4.4**.

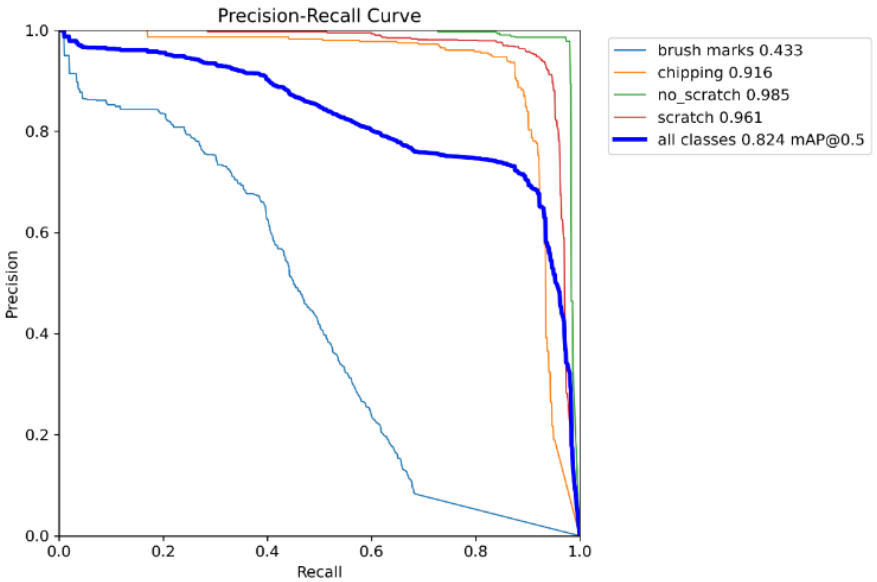
Tabel 4.4 Analisa Recall Confidence-Curve dengan Rata-Rata Recall Confusion Matrix

Metrik Kinerja	**Pelatihan Pertama**	**Pelatihan Kedua**	**Keterangan**
Recall-Confidence Curve	0,89	0,89	Performa dasar model yang stabil di antara kedua pelatihan.
Confidence Threshold	0,00	0,00	Confidence threshold yang digunakan sama.

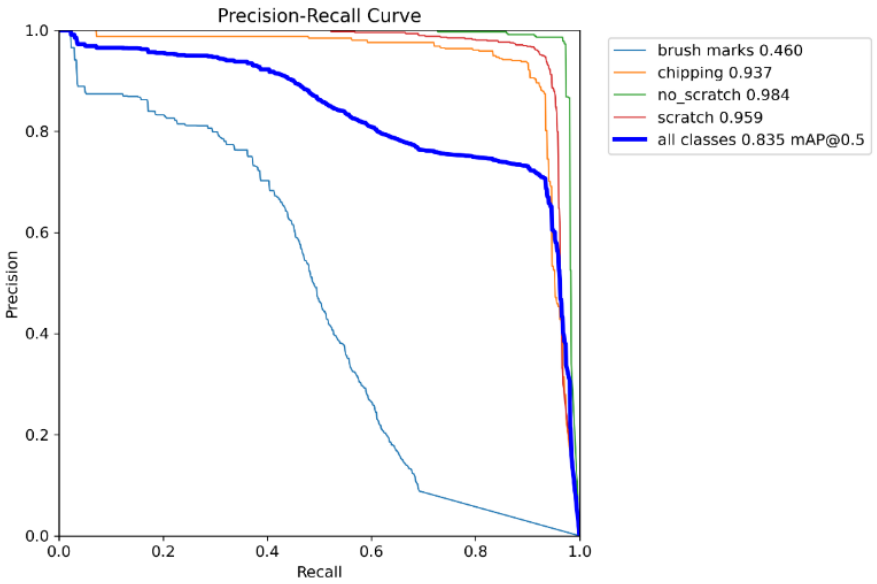
Metrik Kinerja	Pelatihan Pertama	Pelatihan Kedua	Keterangan
Rata-rata Recall Confusion Matrix	0,6851 pada Confidence Threshold ≈ 0,61	0,6669 pada Confidence Threshold ≈ 0,72	Recall confusion matrix menurun namun Confidence Threshold yang digunakan meningkat.
Penurunan Recall	0,89 – 0,6851= 0.2049.	0,89 – 0,6669 = 0.2231	Kenaikan Confidence Threshold dari 0,0 ke 0,61 dan 0,72 membuat recall keduanya turun.

4.1.5. *Precision-Recall curve*

Precision-recall curve adalah kurva yang memvisualisasikan hubungan antara presisi dengan *recall* dari sebuah model. Kurva ini menunjukkan bagaimana presisi berubah seiring dengan perubahan *recall*. Dengan begitu, ambang batas optimal dapat ditentukan untuk memberikan keseimbangan terbaik antara kedua metrik tersebut sesuai dengan kebutuhan pengaplikasian. Untuk dapat melihat kedua kurva tersebut dapat dilihat pada **Gambar 4.9** dan **Gambar 4.10**.



Gambar 4.9 *Precision-Recall Curve* Pelatihan Pertama



Gambar 4.10 *Precision-Recall Curve* Pelatihan Kedua

Berdasarkan kedua kurva *precision-recall* tersebut dapat dianalisis bahwa terdapat kenaikan kinerja model pada pelatihan kedua yang menghasilkan mAP@0,5 sebesar 0,835 untuk semua kelas. Yang dimana hasil ini merupakan peningkatan dari pelatihan pertama yaitu 0,824. Angka yang terletak di samping dama per kelas, seperti *chipping* 0,937 adalah nilai *Average Precision* (AP) spesifik untuk kelas tersebut. Metrik "*all classes* 0,835 mAP@0,5" berarti *mean Average Precision* (mAP) atau rata-rata AP semua kelas adalah 0,835 dengan syarat ambang batas *Intersection over Union* (IoU *threshold*) antara prediksi dan *ground truth* adalah ≥ 0,5 (ambang batas True Positive). IoU Threshold adalah nilai yang tetap dan hanya digunakan untuk menentukan apakah *bounding box* hasil prediksi cukup akurat secara lokasi untuk dihitung sebagai *True Positive* pada titik manapun di kurva. Berbeda dengan *confidence threshold*, *confidence threshold* adalah nilai yang digunakan untuk mengetahui seberapa yakin model dalam menentukan deteksinya per kelas dari 0 sampai 1. Sementara itu, hubungan antara presisi (mengukur seberapa relevan prediksi positif), *recall* (mengukur seberapa banyak deteksi yang dilakukan), dan mAP adalah bahwa kurva *precision-recall* menggambarkan pertukaran keduanya di seluruh *confidence threshold*, dan mAP adalah metrik rata-rata yang merangkum kinerja area di bawah kurva tersebut.

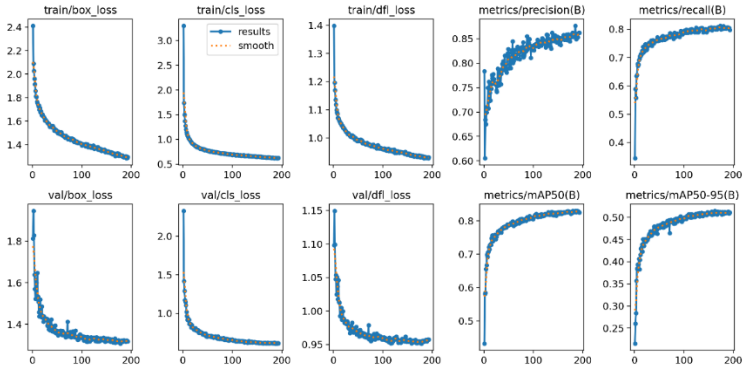
Kurva *Precision-Recall* menunjukkan kinerja presisi dan *recall* model di semua ambang batas *confidence*, berbeda dengan presisi dan *recall* dari *confusion matrix* yang berasal dari sub bab 4.1.1., yang dihitung pada satu titik *confidence threshold*. Pada pelatihan pertama *confusion matrix* menghasilkan presisi 0,6903 dan *recall* 0,6851, apabila nilai tersebut dibawa ke kurva *precision-recall* maka presisi 0,6903 sesuai dengan *recall* di kisaran 0,75 – 0,8 dan *recall* 0,6851 sesuai dengan presisi di kisaran 0,78 – 0,83. Sementara itu, pada pelatihan kedua *confusion matrix* dihasilkan presisi 0,7051 dan *recall* 0,6669. Presisi 0,7051 sesuai dengan *recall* di kisaran 0,75 – 0,8 dan *recall* 0,6669 sesuai dengan presisi di kisaran 0,8 – 0,85. Perbedaan ini disebabkan oleh konsep yang berbeda pada masing-masing evaluasi. Kurva adalah rata-rata seluruh kinerja, sedangkan *confusion matrix* adalah kinerja pada satu titik. Untuk dapat melihat lebih ringkas dapat dilihat pembahasan analisa hasil dalam **Tabel 4.5**.

Tabel 4.5 Analisa *Precision-Recall Curve* dengan *Precision-Recall Confusion Matrix*

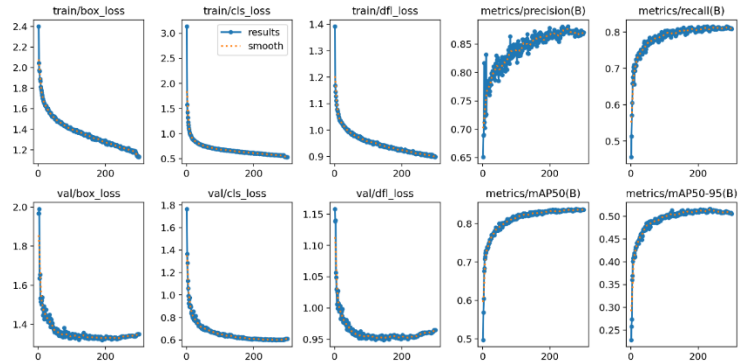
Metrik Kinerja	**Pelatihan Pertama**	**Pelatihan Kedua**	**Keterangan**
mAP@0,5	0,824	0,835	Terjadi kenaikan model keseluruhan sebesar 0,011.
Presisi kurva	0,78 – 0,83	0,8 – 0,85	Terjadi kenaikan presisi sebesar 0,02
Recall kurva	0,75 – 0,83	0,75 – 0,8	Terjadi penurunan recall sebesar 0,03
Kurva dengan confusion matrix	Presisi/Recall Kurva (0.75-0.83) lebih tinggi dari CM (0.68-0.69)	Presisi/Recall Kurva (0.75-0.85) lebih tinggi dari CM (0.66-0.70)	Kurva PR menghitung rata-rata kinerja di semua *Confidence Threshold* (yang berubah-ubah), sementara CM hanya mencerminkan kinerja pada satu titik *Confidence Threshold* (yang tetap).

4.1.6. Kurva Hasil *Training*

Kurva hasil *training* merupakan sekumpulan kurva yang dihasilkan dari pelatihan Google Colaboratory untuk model *object detection*, khususnya dari hasil framework *Ultralytics* YOLOv8. Kurva ini bertujuan untuk memantau pembelajaran model dan mendiagnosis masalah berdasarkan sumbu x (*epochs*) dan sumbu y (*loss value* dan *metrics value*). Untuk dapat melihat kedua kurva hasil *training*/pelatihan dapat dilihat pada **Gambar 4.11** dan **Gambar 4.12**.



Gambar 4.11 Kurva Hasil Pelatihan Pertama



Gambar 4.12 Kurva Hasil Pelatihan Kedua

Berdasarkan kedua gambar tersebut, dihasilkan plot metrik dari dua sesi pelatihan model *Object Detection* yang dilakukan menggunakan arsitektur YOLOv8 selama 200 dan 300 epoch. Kedua percobaan menunjukkan proses pelatihan yang berbeda. Keduanya memiliki kurva *loss* dan *kurva* metrik kinerja. Yang masing-masing akan dibahas di bawah ini

a. Analisis Kerugian (*Loss*)

Loss digunakan untuk mengukur seberapa baik model memprediksi *ground truth* selama pelatihan dan validasi.

1) Kerugian Pelatihan (*Train Loss*)

- *train/box_loss* (kerugian lokalisasi)

  Kerugian untuk penyesuaian *bounding box* pada data *training*. Kedua model menunjukkan penurunan yang stabil, namun pelatihan kedua berhenti pada nilai yang sedikit lebih rendah (≈ 1.2) dibandingkan pelatihan pertama (≈ 1.25).

- *train/cls_loss* (kerugian klasifikasi)

  Kerugian untuk memprediksi kelas objek pada data *training*. Kedua model menunjukkan nilai yang hampir identik dan konvergen (≈ 0.65).

- *train/dfl_loss* (kerugian *distribution focal loss*)

  Kerugian Distribution Focal Loss atau sejenisnya, terkait dengan penajaman prediksi batas. Kedua model menunjukkan konvergensi yang mirip.

2) Kerugian Validasi (*Validation Loss*)

- val/box_loss

  Kerugian untuk penyesuaian *bounding box* pada data validasi. Pada pelatihan pertama menunjukkan nilai 1,3 dan pelatihan kedua menunjukkan nilai 0,7. Hal ini membuktikan bahwa model pada pelatihan kedua lebih unggul dalam melokalisasi objek pada data baru.

- val/cls_loss

  Kerugian untuk memprediksi kelas objek pada data validasi. Pada pelatihan kedua juga menunjukkan performa yang lebih baik daripada pelatihan pertama yakni 0,65 dan 0,7.

b. Analisis Metrik Kinerja (mAP, *Precision*, *Recall*)

Metrik ini mengukur kualitas deteksi model secara keseluruhan menggunakan dataset validasi.

1) Dasar Penamaan

Pada setiap akhir penamaan metrik kinerja diberikan kode huruf B. Hal ini merujuk pada penggunaan *framework Ultralytics* YOLOv8 yaitu B untuk *Bounding Box* yang merupakan keluaran dari projek *object detection*.

2) Perbandingan Metrik
   - *metrics/precision* (B) dan *metrics/recall*

     Metrik untuk mengukur kemampuan presisi dan *recall* suatu model. Pada pelatihan kedua presisi mencapai 0,88 dan *recall* 0,82 yang sedikit melampaui pelatihan pertama yakni presisi 0,86 dan *recall* 0,8. Hal ini menunjukkan bahwa pelatihan kedua memiliki keseimbangan yang sedikit lebih baik antara meminimalkan *False Positive* (presisi tinggi) dan meminimalkan *False Negative* (*recall* tinggi).
   - metrics/mAP50 (B)

     Metrik untuk mengukur *Mean Average Precision* pada ambang batas IoU 0,5. Pada pelatihan kedua menunjukkan nilai 0,9 dan pelatihan pertama sebesar 0,88 dimana membuktikan bahwa pelatihan kedua memiliki hasil yang sedikit lebih baik dalam prediksi yang dianggap benar (*True Positive*).
   - *metrics*/mAP50-95 (B)

     Metrik untuk mengukur kualitas model dalam *Mean Average Precision* yang memiliki IoU dengan ambang batas 0,50 sampai 0,95. Pada pelatihan kedua menunjukkan nilai 0,52 dan pelatihan pertama sebesar 0,5 dimana membuktikan bahwa pelatihan kedua memiliki hasil yang lebih baik dalam menemukan dan menggambarkan objek dengan lokalisasi *bounding box* IoU tinggi.

Dengan melihat dan menganalisis hasil kedua kurva *training* tersebut dihasilkan model pelatihan kedua yang lebih unggul. Kemampuan lokalisasi model yang ditunjukkan oleh nilai rendah *val*/*box_loss* dan mAP50-95 (B) yang lebih tinggi. Sehingga, model pelatihan kedua merupakan model yang dipakai karena memiliki kemampuan generalisasi dan akurasi deteksi yang lebih baik dari

pelatihan pertama. Untuk dapat melihat performa model secara ringkas dapat dilihat pada **Tabel 4.6**.

Tabel 4.6 Analisa Kedua Kurva Hasil *Training* Model

Metrik	Pelatihan Pertama	Pelatihan Kedua	Keterangan
train/box_loss	1,25	1,2	Pelatihan kedua lebih baik dalam training penyesuaian *bounding box*
train/cls_loss	0,65	0,65	Kedua pelatihan sama baiknya dalam training penentuan kelas
val/box_loss	1,3	0,7	Pelatihan kedua lebih baik dalam validasi penyesuaian *bounding box*
val/cls_los	0,7	0,65	Pelatihan kedua sedikit lebih baik dalam validasi *training* penentuan kelas
mAP50(B)	0,88	0,90	Pelatihan kedua memperoleh rata-rata presisi seluruh kelas yang lebih baik
mAP50-95(B)	0,50	0,52	

4.2.Pengujian dan Analisa Optimasi Model Terhadap FPS

Pada bagian ini dilakukan analisis terhadap performa model deteksi kecacatan dari sisi *Frames Per Second* (FPS) yang dihasilkan. Analisis ini berfokus pada upaya optimasi model untuk memperoleh FPS yang lebih tinggi tanpa mengorbankan akurasi deteksi secara berlebihan. Terdapat tiga pendekatan yang dilakukan untuk menguji performa model, yaitu penggunaan model YOLOv8 secara langsung, konversi model menggunakan NCNN, dan penambahan *inference interval* pada NCNN dengan masing-masing dilakukan 12 kali pengujian.

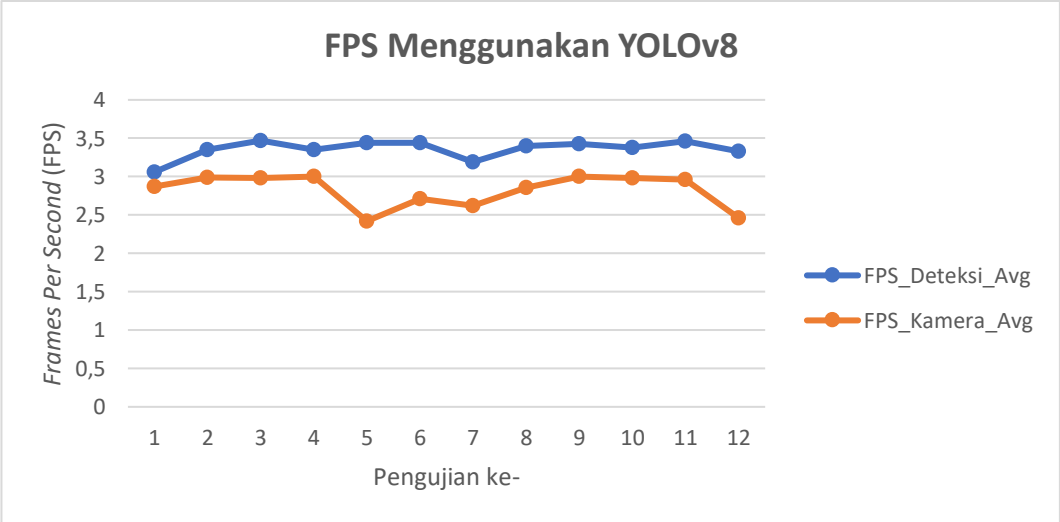
4.2.1. FPS Penggunaan YOLOv8

Pengujian ini dilakukan menggunakan model pelatihan YOLOv8 yang telah ditentukan sebelumnya pada sub bab 4.1.6. yakni menggunakan model YOLOv8 pelatihan kedua. Pengujian berfungsi sebagai acuan dasar untuk melihat performa

awal model sebelum dilakukan optimasi lebih lanjut. Untuk dapat melihat FPS dalam penggunaan YOLOv8, ditunjukkan tabel beserta grafik pada **Tabel 4.7** dan **Gambar 4.13**.

Tabel 4.7 FPS Penggunaan YOLOv8

Pengujian ke -	Data Pengujian	FPS_Deteksi_Avg	FPS_Kamera_Avg
1	A	3,06	2,62
2	B	3,35	2,71
3	C	3,47	2,42
4	D	3,35	3
5	E	3,44	2,98
6	F	3,44	2,99
7	G	3,19	2,87
8	H	3,4	2,86
9	I	3,43	3
10	J	3,38	2,98
11	K	3,46	2,96
12	L	3,33	2,46
AVERAGE FPS		**3,35**	**2,82**



Gambar 4.13 Grafik FPS Penggunaan YOLOv8

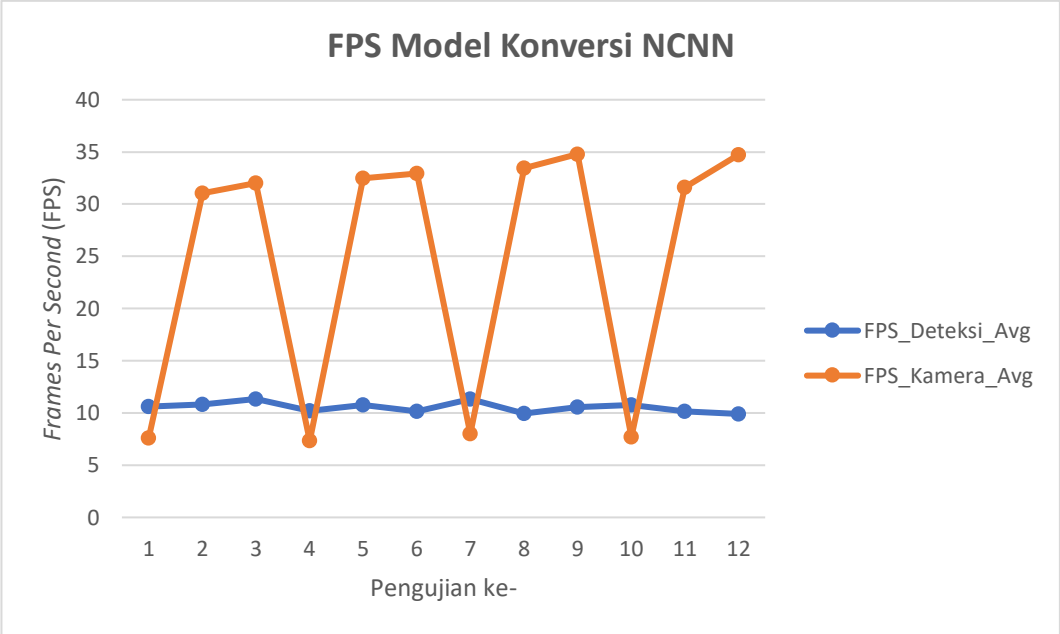
Berdasarkan hasil pengujian terhadap 12 data pengujian menunjukkan bahwa rata-rata FPS_Deteksi sebesar 3,35 FPS, sedangkan FPS_Kamera sebesar 2,82 FPS. Angka ini masih jauh dari kebutuhan deteksi secara real-time (sekitar 15 FPS). FPS_Kamera yang lebih rendah dari FPS_Deteksi juga mengindikasikan bahwa sistem secara keseluruhan lambat, baik dalam proses akuisisi citra maupun deteksi. Sehingga, dapat dikatakan bahwa penggunaan model YOLOv8 murni belum mampu memenuhi performa yang diharapkan.

4.2.2. FPS Model Konversi NCNN

Pengujian selanjutnya model YOLOv8 dikonversi ke format NCNN. Pengujian dilakukan untuk mengetahui seberapa besar pengaruh kuantisasi model dalam perubahan FPS. Untuk dapat melihat FPS dalam penggunaan konversi model YOLOv8 menjadi NCNN ditunjukkan pada **Tabel 4.8** dan **Gambar 4.14**.

Tabel 4.8 FPS Model Konversi NCNN

Pengujian ke-	Data Pengujian	FPS_Deteksi_Avg	FPS_Kamera_Avg
1	A	10,62	7,6
2	B	10,81	31,02
3	C	11,34	32
4	D	10,21	7,33
5	E	10,75	32,49
6	F	10,14	32,93
7	G	11,33	8
8	H	9,96	33,47
9	I	10,57	34,79
10	J	10,79	7,7
11	K	10,15	31,6
12	L	9,91	34,71
AVERAGE FPS		**10,54**	**24,47**



Gambar 4.14 Grafik FPS Model Konversi NCNN

Berdasarkan hasil pengujian konversi model YOLOv8 menjadi NCNN diperoleh hasil yang membaik. Rata-rata FPS_Deteksi yang diperoleh sebesar 10,54 FPS atau lebih dari tiga kali lipat dibandingkan dengan model YOLOv8 murni. Selain itu, FPS_Kamera meningkat menjadi 24,47 FPS yang berarti proses penangkapan citra kamera berjalan hampir delapan kali lebih cepat berkat kuantisasi model. Peningkatan FPS ini membuktikan bahwa konversi model YOLOv8 ke NCNN meningkat secara signifikan sebesar 466,72%.

4.2.3. FPS Model Konversi NCNN dan *Infer Interval*

Pengujian selanjutnya dilakukan optimasi lanjutan dengan menambahkan *inference interval* pada pemrosesan model NCNN. Pengujian dilakukan untuk menurunkan frekuensi inferensi per detik, sehingga *resource* komputasi CPU tidak perlu memproses setiap frame yang masuk. Hal ini berpotensi untuk meningkatkan FPS kamera. Untuk mengetahui pengaruhnya, dilakukan dua pengujian dengan interval yang berbeda, yaitu 0,15 detik dan 0,2 detik.

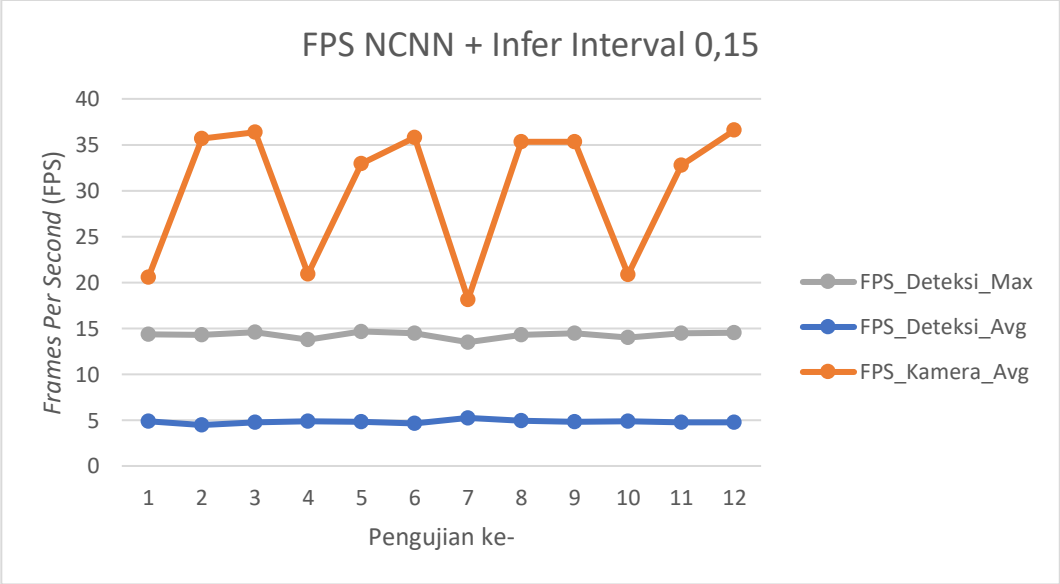
a. NCNN dan *Infer Interval* 0,15

Pada pengujian ini, *infer interval* ditetapkan sebesar 0,15 detik, artinya Raspberry Pi 5 hanya melakukan satu kali proses inferensi setiap 0,15

detik. Untuk dapat mengetahui pengaruhnya, disajikan tabel dan grafik pada **Tabel 4.9** dan **Gambar 4.15**.

Tabel 4.9 NCNN dan *Infer Interval* 0,15

Pengujian ke-	Data	FPS_Deteksi_Max	FPS_Deteksi_Avg	FPS_Kamera_Avg
1	A	14,36	4,9	20,56
2	B	14,27	4,46	35,68
3	C	14,58	4,74	36,35
4	D	13,75	4,89	20,89
5	E	14,66	4,8	32,95
6	F	14,47	4,62	35,76
7	G	13,49	5,24	18,15
8	H	14,27	4,93	35,31
9	I	14,47	4,8	35,35
10	J	14	4,85	20,87
11	K	14,48	4,77	32,76
12	L	14,5	4,75	36,58
AVERAGE FPS		**14,27**	**4,81**	**30,1**



Gambar 4.15 Grafik FPS NCNN dan *Infer Interval* 0,15

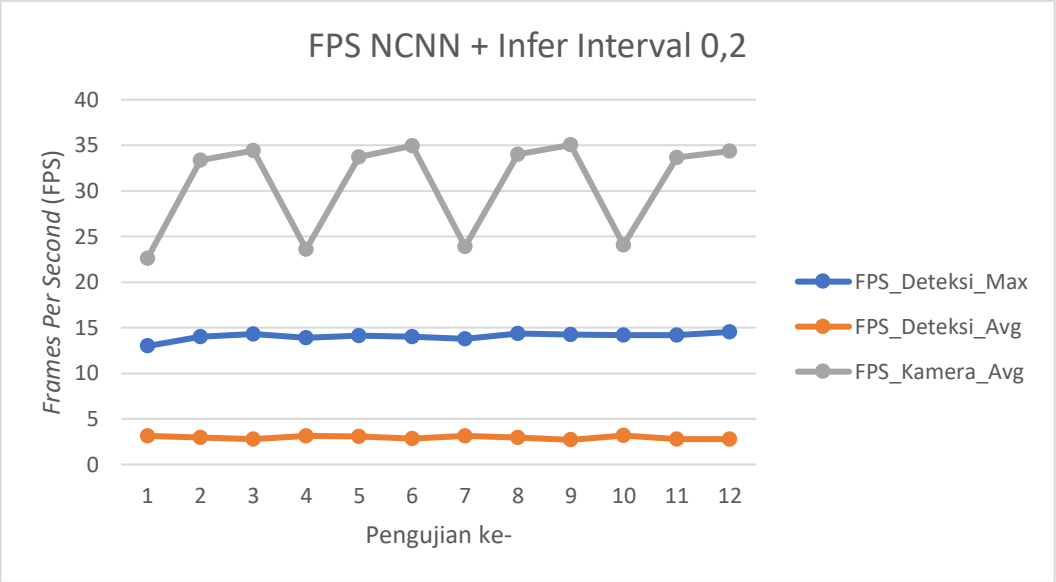
Pada pengujian ini angka FPS_Deteksi_Max mencapai 14,27 yang mana hampir mencapai target *real-time* 15 FPS. Kenaikan rata-rata FPS_Deteksi_Max adalah 325,12% dari YOLOv8 (baseline). Namun, FPS_Deteksi_Avg turun menjadi 4,81 FPS dari FPS_Deteksi NCNN yaitu 10,55 FPS. Hal ini disebabkan oleh *inference interval* yang secara sengaja mengurangi jumlah inferensi yang dilakukan dalam satu detik. Sehingga pemrosesan frame dan stabilitas lebih diprioritaskan untuk menghasilkan FPS_Kamera yang tinggi.

b. NCNN dan *Infer Interval* 0,2

Pengujian berikutnya dilakukan dengan *infer interval* sebesar 0,2 detik. Untuk dapat melihat hasil performa model dengan jeda yang sedikit lebih lama antar proses inferensi, disajikan tabel dan grafik pada **Tabel 4.10** dan **Gambar 4.16**.

Tabel 4.10 NCNN dan *Infer Interval* 0,2

Pengujian ke	**Data**	**FPS_Deteksi_Max**	**FPS_Deteksi_Avg**	**FPS_Kamera_Avg**
1	A	12,99	3,11	22,63
2	B	14,03	2,96	33,37
3	C	14,28	2,8	34,41
4	D	13,91	3,15	23,6
5	E	14,13	3,08	33,72
6	F	14,03	2,86	34,98
7	G	13,78	3,12	23,91
8	H	14,35	2,98	34,04
9	I	14,26	2,69	35,04
10	J	14,17	3,17	24,1
11	K	14,19	2,78	33,67
12	L	14,53	2,77	34,37
AVERAGE FPS		**14,05**	**2,95**	**30,65**



Gambar 4.16 Grafik FPS NCNN dan *Infer Interval* 0,2

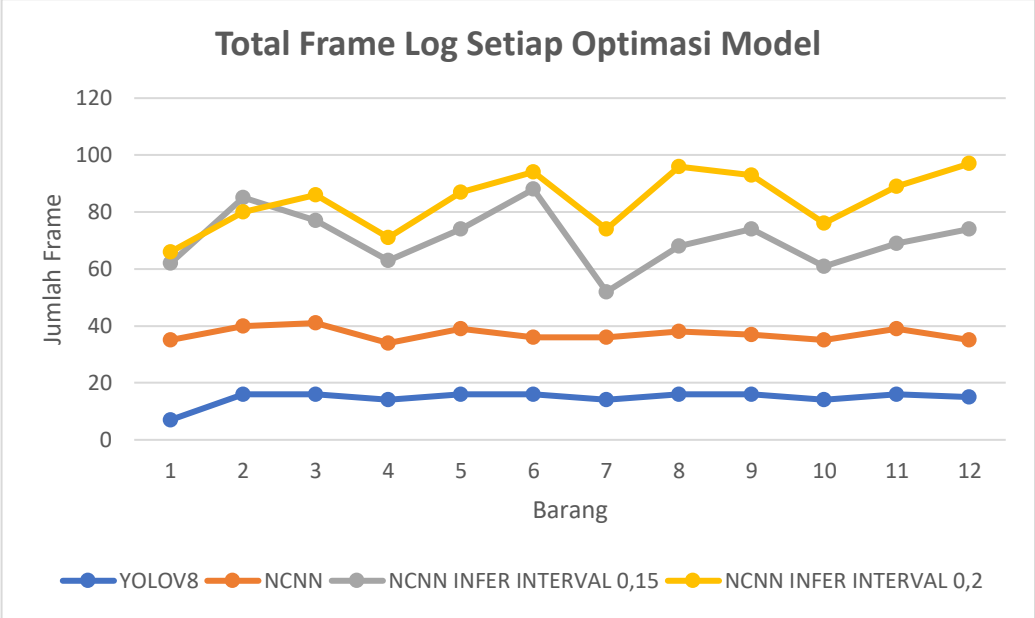
Pada pengujian ini dihasilkan FPS_Deteksi_Max yang sedikit menurun menjadi 14,05 FPS namun masih sangat dekat dengan target. Peningkatan rata-rata FPS_Deteksi_Max adalah 318,59% dari *baseline.* Sama seperti sebelumnya, FPS_Deteksi_Avg menurun menjadi 2,96 FPS karena frekuensi inferensi yang lebih rendah. FPS_Kamera menghasilkan rata-rata 30,65 yang mengindikasikan bahwa interval inferensi yang sedikit lebih lama dapat mengurangi kinerja CPU untuk meningkatkan pengambilan dan pemrosesan frame.

4.2.4. *Total Frame Log* Deteksi Setiap Optimasi Model

Total frame log deteksi yaitu jumlah keseluruhan *frame* deteksi model yang berhasil diproses kemudian dicatat (log) oleh sistem selama periode pengujian deteksi setiap barang atau dari saat kamera mulai mendeteksi hingga keputusan layak atau cacat diambil. Nilai ini juga berkaitan erat dengan FPS dan durasi deteksi yang dilakukan. Durasi deteksi yang dilakukan pada setiap barang adalah selama 5 detik per barang. Untuk dapat melihat lebih jelasnya disediakan **Tabel 4.11** dan **Gambar 4.17** dibawah.

Tabel 4.11 *Total Frame Log* Setiap Optimasi Model

Barang	YOLOv8	NCNN	NCNN Infer Interval 0,15	NCNN Infer Interval 0,2
1	7	35	62	66
2	16	40	85	80
3	16	41	77	86
4	14	34	63	71
5	16	39	74	87
6	16	36	88	94
7	14	36	52	74
8	16	38	68	96
9	16	37	74	93
10	14	35	61	76
11	16	39	69	89
12	15	35	74	97
Avg Frame Log	**14,66**	**37,08**	**70,58**	**84,08**



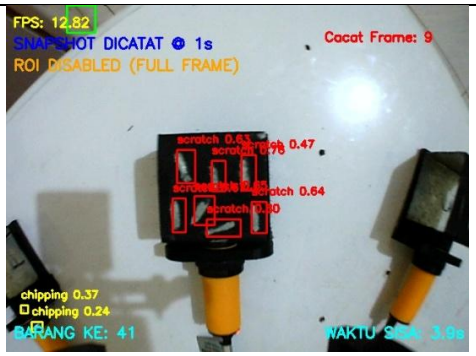
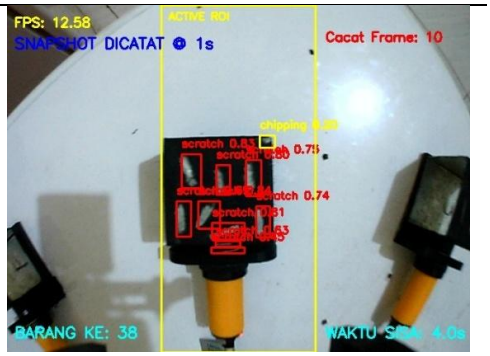
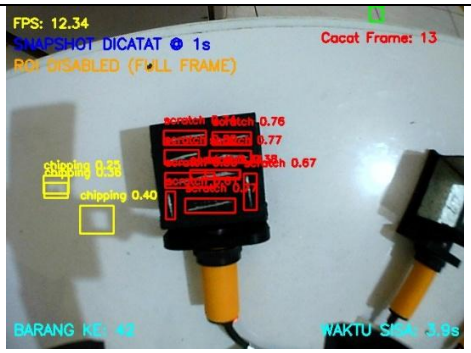
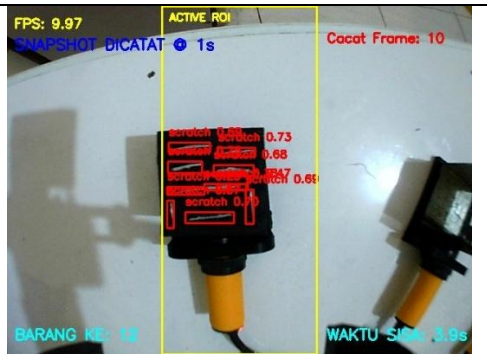
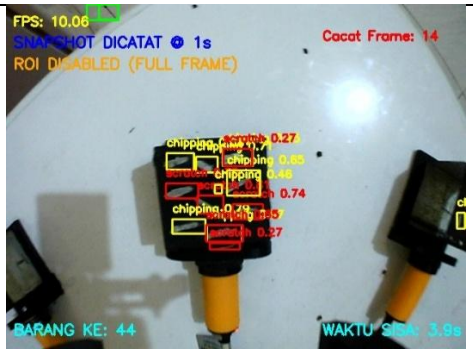
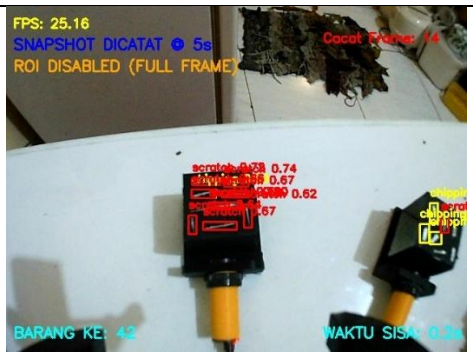
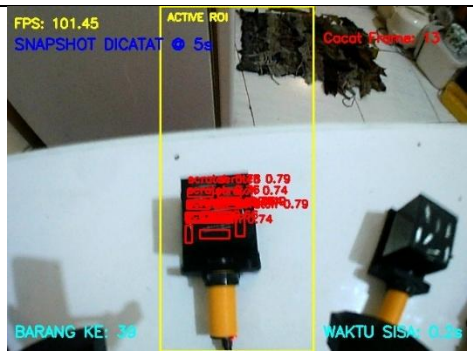
Gambar 4.17 Grafik *Total Frame Log* Setiap Optimasi Model

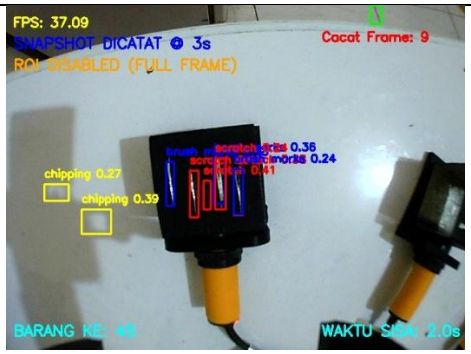
Berdasarkan **Tabel 4.11** dapat diperoleh rata-rata FPS_Deteksi Implisit yang dihitung dari data *Frame Log* dibagi waktu deteksi (5 detik) dan akan dibandingkan dengan FPS_Deteksi_Avg dari pembahasan sebelumnya. Dengan begitu diperoleh nilai rata-rata FPS_Deteksi implisit untuk YOLOv8, konversi NCNN, NCNN *infer interval* 0,15, dan NCNN *infer interval* 0,2 secara berurutan adalah 2,93 FPS, 7,42 FPS, 14,12 FPS, dan 16,82 FPS. Untuk YOLOv8 dan NCNN menunjukkan kenaikan FPS_Deteksi_Avg (3,36 dan 10,55 FPS) daripada FPS_Deteksi implisit (2,93 dan 7,42 FPS). Namun, pada NCNN *infer interval* pola tersebut berbalik. Hal ini disebabkan oleh penurunan FPS_Deteksi_Avg (4,81 dan 2,96 FPS) secara sengaja oleh *infer interval* karena frekuensi inferensi dibatasi. FPS_Deteksi implisit *infer interval* (14,12 dan 16,82 FPS) terutama pada interval 0,2s menunjukkan bahwa dalam durasi 5 detik jumlah frame yang berhasil di log melebihi target 15 FPS. Hal ini menunjukkan bahwa *inference interval* yang lebih jarang (0,2s) memberikan waktu pemrosesan yang cukup bagi sistem untuk mengkompensasi dan memastikan *logging* berjalan setelah setiap inferensi dilakukan. Oleh karena itu, konfigurasi NCNN dengan *Infer Interval* 0,2 detik adalah yang paling efektif dalam memproses jumlah *frame* deteksi yang berhasil diproses (log) mencapai atau melebihi batas *real-time* (15 FPS), menghasilkan rata-rata 84 *frame* deteksi dalam durasi 5 detik pengujian.

4.3. Pengujian dan Analisa Pengaruh Filter *Region Of Interest* (ROI) Pada Kinerja Deteksi

Pada pengujian ini melibatkan penerapan filter *Region of Interest* (ROI) pada proses deteksi kecacatan produk. Penerapan filter ini bertujuan untuk membatasi area deteksi hanya pada bagian tertentu dari citra yang diinginkan sehingga dapat mengurangi *noise* atau gangguan dari area lain yang tidak diperlukan seperti area latar belakang. Untuk dapat mengetahui pengaruh penggunaan filter ROI, sebagaimana ditunjukkan pada **Tabel 4.12**.

Tabel 4.12 Pengaruh Penggunaan Filter ROI

No.	Tanpa Filter ROI	Dengan Filter ROI
1		
2		
3		
4		

No.	Tanpa Filter ROI	Dengan Filter ROI
5		

Berdasarkan data pada **Tabel 4.12**, terlihat bahwa deteksi tanpa Filter ROI cenderung menyebar atau menjalar ke luar objek yang ditargetkan (misalnya, area latar belakang). Kondisi ini dapat menyebabkan deteksi kecacatan yang berlebihan (positif palsu) dan tidak akurat terhadap kondisi produk yang sebenarnya. Data deteksi berdasarkan tabel tersebut diperoleh pengujian tanpa filter ROI menghasilkan total 59 deteksi dari total 40 kecacatan asli, mencatatkan total absoluter error sebesar 19. Ini setara dengan persentase *false detection* atau error pada deteksi tanpa filter sebesar 47,5%.

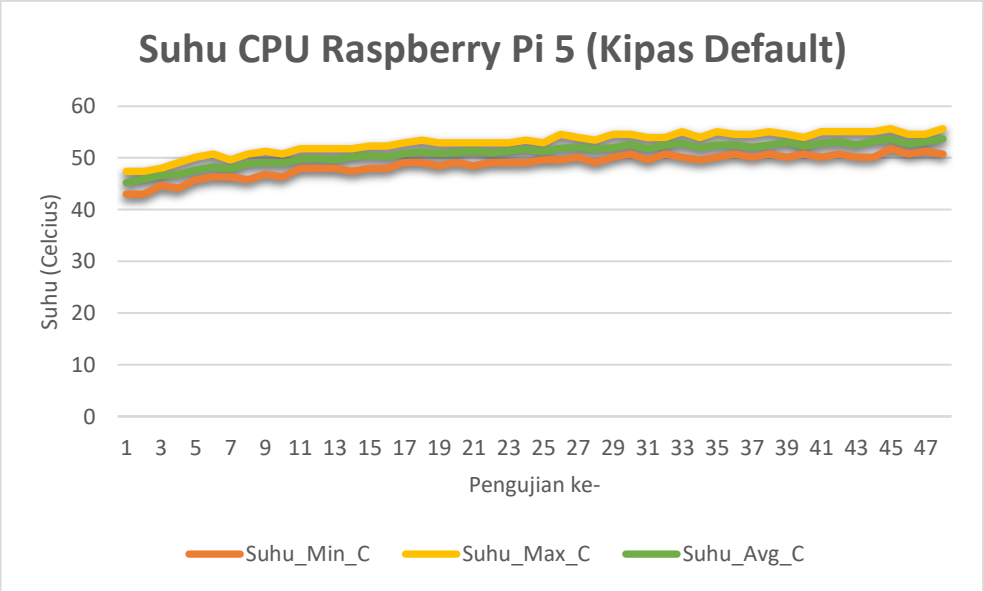
Sebaliknya, setelah filter ROI diimplementasikan, total deteksi menurun menjadi 51 dengan total *absolute error* sebesar 11, sehingga persentase *false detection* berhasil diturunkan menjadi 27.5%. Perbandingan ini menunjukkan bahwa penggunaan filter ROI mampu menekan tingkat *error* sistem deteksi. Secara mutlak, terjadi penurunan persentase *false detection* sebesar 20%. Oleh karena itu, Filter ROI diterapkan untuk memfokuskan area deteksi. Filter ini bekerja dengan cara memusatkan proses deteksi hanya pada posisi tengah atau grid tengah dari format grid 3x1 yang telah ditentukan, memastikan objek yang dideteksi menjadi terisolasi dan terfokus pada objek utama saja.

4.4.Pengujian dan Analisa Pengaruh Sistem Pendingin Terhadap Kinerja Raspberry Pi 5

Pada pengujian ini digunakan konfigurasi model yang telah di konversi menjadi NCNN ditambah *infer interval* 0,2 dan filter ROI. Pengujian ini dilakukan untuk mengetahui pengaruh kondisi suhu terhadap performa Raspberry Pi 5. Pengujian dibedakan menjadi 2 yakni pengaturan kipas secara *default* dan dimodifikasi *user*.

a. Suhu CPU pada RPM Kipas *Default*

Kipas yang ditancapkan pada Raspberry Pi 5 secara *default* memiliki pemrograman yang diatur menyala RPM sedang pada 60 derajat celcius dan menyala full RPM pada 70 derajat celcius. Berikut dibawah ini **Gambar 4.18** merupakan grafik kenaikan suhu Raspberry Pi 5.



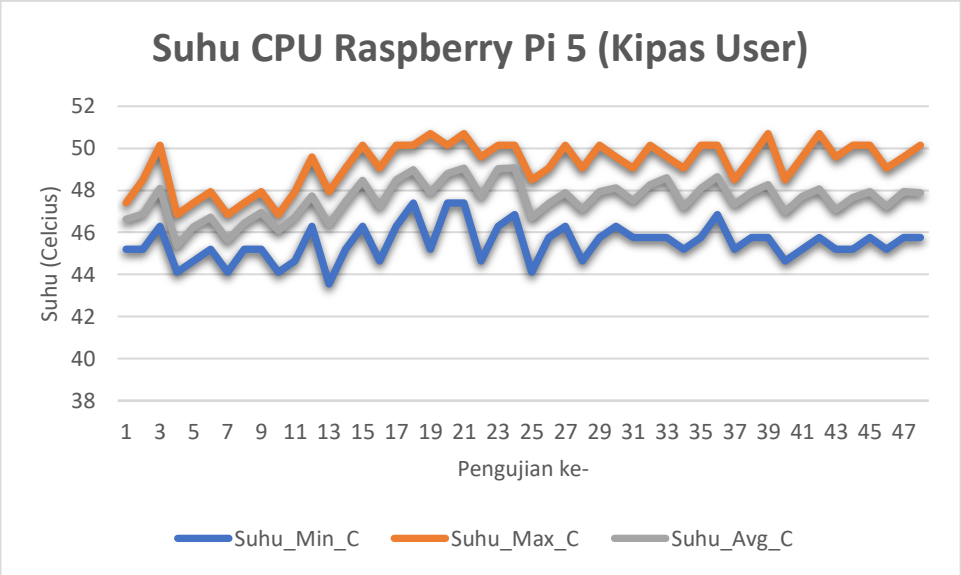
Gambar 4.18 Grafik Suhu CPU Raspberry Pi 5 dengan RPM Kipas Default

b. Suhu CPU pada RPM Kipas *User*

Untuk dapat mengubah RPM kipas sesuai dengan keinginan diperlukan modifikasi pada /boot/firmware/config.txt agar saat suhu mencapai ≤ 40 derajat celcius, kipas mati dan saat suhu mencapai ≥ 45 derajat celcius, kipas menyala. Berikut dibawah ini **Gambar 4.19** merupakan modifikasi dan grafik suhu pada **Gambar 4.20**.

```
 GNU nano 7.2                  /boot/firmware/config.txt

[cm4]
# Enable host mode on the 2711 built-in XHCI USB controller.
# This line should be removed if the legacy DWC2 controller is required
# (e.g. for USB device mode) or if USB support is not required.
otg_mode=1

[cm5]
dtoverlay=dwc2,dr_mode=host

[all]
dtoverlay=dwc2
enable_uart=1

# Kipas akan mati  pada 40C
dtparam=fan_temp0=40000,fan_temp0_speed=0

# Kipas akan menyala saat suhu naik sama dengan atau di atas dari 45C
dtparam=fan_temp1=45000,fan_temp1_speed=255
```

Gambar 4.19 Pengaturan dalam /boot/firmware/config.txt



Gambar 4.20 Grafik Suhu CPU Raspberry Pi 5 dengan RPM Kipas User

Dengan kedua konfigurasi diatas dihasilkan kinerja CPU Raspberry Pi 5 saat menjalankan model NCNN serta *infer interval* dibawah pengaruh sistem pendingin yaitu pengaturan kipas *default* dan *user*. Berdasarkan data yang diperoleh pada **Tabel 4.13**, konfigurasi kipas *user* menunjukkan efektivitas pendinginan yang berhasil menurunkan suhu rata-rata dari 50,93℃ (*default*) menjadi 47,57℃ dan suhu maksimal dari 52,93℃ menjadi 49,24℃ atau terjadi penurunan lebih dari 3℃. Meskipun terjadi penurunan suhu yang signifikan, data frekuenci CPU tidak menunjukkan adanya peningkatan kinerja yang berdampak. Frekuensi rata-rata CPU pada kedua skenario sangat identik dan mendekati batas maksimal sistem (2,4 GHz), yaitu 2,398 GHz untuk Default dan 2,397 GHz untuk *user*. Hal ini menunjukkan bahwa Raspberry Pi 5 di bawah pengaturan kipas *default* belum mencapai ambang batas *thermal throttling*, sehingga pendinginan yang lebih agresif (*user*) hanya meningkatkan margin keamanan termal tanpa secara langsung meningkatkan frekuensi pemrosesan. Sehingga, penggunaan pendinginan *user* dipilih agar memberikan lingkungan operasional yang lebih dingin yang baik untuk stabilitas jangka panjang dan umur komponen meskipun tidak memberikan boost kinerja yang signifikan.

Tabel 4.13 Perbandingan Pengaruh Suhu Terhadap Performa CPU Raspberry Pi 5

Pengaturan RPM Kipas	CPU Min	CPU Max	CPU Avg	Suhu Min	Suhu Max	Suhu Avg
default	2,2	2,4	2,398	48,7	52,93	50,93
user	2,26	2,4	2,397	45,5	49,24	47,57

4.5.Pengujian dan Analisa Hasil Deteksi Kecacatan Produk

Untuk menguji kinerja sistem deteksi kecacatan produk dilakukan serangkaian pengujian dengan berbagai sampel data pengujian. Pengujian dilakukan sejumlah 12 kali dengan identifikasi jenis-jenis kecacatan yaitu *scratch* (goresan), *brush marks* (bekas kuas), dan *Chipping* (cat terkelupas). Berikut dibawah ini **Tabel 4.14** merupakan hasil lengkap dari 12 kali pengujian yang dilakukan pada berbagai sampel data pengujian.

Tabel 4.14 Pengujian Deteksi Kecacatan Produk

Pengujian Ke -	Data	Scratch	Brush Marks	Chipping	No Scratch	Total	Status
1	A	0	0,1	0	0,05	0,1	Layak
	B	6	0	1,1	0	7,1	Cacat
2	B	5,1	0	1	0,19	6,1	Cacat
	C	2,24	0	0,05	0,05	2,29	Cacat
3	C	3,33	0	0,05	0,05	3,38	Cacat
	D	0	0,05	0,05	0,05	0,1	Layak
4	D	0	0	0	0,15	0	Layak
	E	2,19	0	1,38	0,1	3,57	Cacat
5	E	1	0	0,68	0,42	1,68	Cacat
	F	6,26	0	0,11	0	6,37	Cacat
6	F	4,86	0	0	0,1	4,86	Cacat
	G	0	0,57	0,1	0	0,67	Layak
7	G	0	0,15	0	0,3	0,15	Layak
	H	0,9	0	6,55	0,05	7,45	Cacat

Pengujian Ke -	Data	Scratch	Brush Marks	Chipping	No Scratch	Total	Status
8	H	1,8	0	4,4	0	6,2	Cacat
	I	0,15	0,15	0,1	0,05	0,4	Layak
9	I	1,65	0,2	0,05	0,05	1,9	Cacat
	J	0	0,68	0,05	0	0,74	Layak
10	J	0	0	0	0,1	0	Layak
	K	0,25	0	3,05	0,1	3,3	Cacat
11	K	0,05	0,05	2,65	0,25	2,75	Cacat
	L	2,95	0	1,45	0,05	4,4	Cacat
12	L	3,6	0,05	0,95	0,6	4,6	Cacat
	A	0	2,67	0,05	0,05	2,71	Cacat

Angka-angka pada **Tabel 4.14** tersebut berasal dari nilai rata-rata dari jumlah kecacatan yang berhasil dideteksi oleh sistem pada setiap frame log deteksi selama satu kali proses pengujian per produk/barang. Berdasarkan sub bab 4.2.4, rata-rata jumlah *frame log* yang dihasilkan setiap kali mendeteksi satu barang adalah 84,083 *frame*. Apabila produk yang ingin di deteksi memiliki 3 kecacatan, maka dalam setiap satu *frame* harus terdapat 3 deteksi cacat tersebut. Sehingga, pada perhitungan total jumlah deteksi kecacatan pada seluruh *frame log* dibagi dengan jumlah *frame log* akan menghasilkan deteksi cacat berjumlah 3 yang mana sesuai dengan cacat produk asli. Dengan begitu, pada pengujian ini penulis membuat kriteria bahwa produk dinyatakan cacat jika nilai deteksi kecacatan sama dengan atau di atas 1. Sebaliknya, nilai dibawah 1 dianggap sebagai produk yang layak.

Berdasarkan **Tabel 4.14**, setiap sampel produk diuji pada dua posisi pengambilan citra yang berbeda, yaitu posisi 1 dan posisi 2, dengan tujuan untuk menganalisis pengaruh perpindahan posisi terhadap hasil deteksi kecacatan produk. Hasil pengujian menunjukkan bahwa sebagian besar sampel menghasilkan status klasifikasi yang konsisten antara kedua posisi, yang menandakan bahwa sistem deteksi kecacatan memiliki tingkat kestabilan yang cukup baik. Namun demikian,

masih ditemukan beberapa sampel yang mengalami perubahan status akibat perbedaan kondisi pengujian pada masing-masing posisi.

Sampel yang menunjukkan ketidakkonsistenan hasil adalah sampel A dan I. Pada sampel A, produk diklasifikasikan sebagai *Layak* pada posisi 1, namun berubah menjadi *Cacat* pada posisi 2. Perubahan ini disebabkan oleh meningkatnya nilai *brush marks* yang terdeteksi pada posisi 2 sehingga total kecacatan melampaui ambang batas kelayakan. Hal ini menunjukkan bahwa pola bekas kuas pada permukaan produk menjadi lebih terlihat pada kondisi pengambilan citra tertentu. Sementara itu, pada sampel I, perbedaan status terjadi akibat meningkatnya nilai *scratch* yang terdeteksi pada salah satu posisi pengujian, yang mengindikasikan bahwa kecacatan pada permukaan produk tidak terdistribusi secara merata dan hanya terdeteksi secara optimal pada posisi tertentu.

Ketidakkonsistenan hasil deteksi tersebut tidak hanya dipengaruhi oleh perbedaan sudut pengambilan citra, tetapi juga oleh faktor lain seperti pencahayaan atau intensitas cahaya ruangan, jarak kamera terhadap objek, serta kualitas citra yang dihasilkan. Variasi pencahayaan dapat menyebabkan bayangan, pantulan cahaya (*glare*), dan perubahan kontras pada permukaan hollow galvanis steel berwarna hitam, sehingga memengaruhi kemampuan model dalam mengenali pola kecacatan. Selain itu, perbedaan jarak kamera mengakibatkan perubahan skala objek dan ketajaman citra, yang berdampak pada detail kecacatan yang dapat ditangkap oleh sistem.

Faktor lainnya yang turut memengaruhi hasil pengujian meliputi fokus kamera, ketepatan area *Region of Interest* (ROI), ketidakstabilan posisi objek selama pengujian, serta noise citra akibat kondisi lingkungan. Selain itu, keterbatasan jumlah dataset dan kurangnya keragaman data latih juga berpengaruh terhadap performa model. Dataset yang belum cukup besar dan belum mencakup variasi kondisi nyata, seperti perbedaan pencahayaan, sudut pengambilan citra, jarak kamera, serta variasi permukaan material, menyebabkan model belum sepenuhnya robust. Kondisi ini membuat model lebih sensitif terhadap perubahan posisi dan lingkungan pengujian, sehingga pada beberapa sampel terjadi perbedaan hasil klasifikasi antara posisi 1 dan posisi 2.

Dari total 12 sampel yang diuji, sebanyak 10 sampel menunjukkan hasil yang konsisten, sedangkan 2 sampel mengalami perubahan status akibat perpindahan posisi pengujian. Dengan demikian, diperoleh nilai akurasi pengukuran berdasarkan konsistensi status *Layak* dan *Cacat* sebesar 83,33%, dengan error pengukuran sebesar 16,67%. Hasil ini menunjukkan bahwa sistem deteksi kecacatan telah bekerja dengan cukup baik, namun masih dipengaruhi oleh variasi kondisi pengujian dan keterbatasan dataset. Dokumentasi visual selama proses pengujian pada kedua posisi dapat dilihat pada **Lampiran 9**.

4.6. Analisa Uji Penggunaan UART Sebagai Komunikasi Serial

4.6.1. Analisis *Latency* Pengiriman Data

Analisis ini bertujuan untuk mengukur waktu tunda (*latency*) dari deteksi cacat hingga perintah sortir dikirim ke STM32 menggunakan UART sebagai *backbone* komunikasi serial antara Raspberry Pi 5 dan mikrokontroler STM32. Data yang dikumpulkan meliputi Waktu Kirim Serial dan Latensi Total Sistem, yang akan digunakan untuk mengidentifikasi tingkat efisiensi protokol UART serta pengaruh implementasi *delay* sistem terhadap responsivitas waktu nyata. Berikut **Tabel 4.15** dibawah ini merupakan data rata-rata *latency* setiap pengiriman data yang dilakukan.

Tabel 4.15 Rata-rata Latensi Pengiriman Data Setiap Pengiriman Data

	Pengiriman Pertama	**Pengiriman Kedua**
Waktu Kirim Serial (ms)	14,96	14,78
Latensi Total Sistem (ms)	5018,49	20039,43

Berdasarkan **Tabel 4.15**, pengujian dilakukan untuk mengukur waktu tunda (latency) sistem sejak proses deteksi kecacatan pada Raspberry Pi 5 hingga perintah sortir dikirimkan ke mikrokontroler STM32 melalui komunikasi serial UART. Sistem dirancang dengan delay yang disengaja, yaitu waktu tunggu 5 detik sebelum pengiriman data pertama dan 15 detik sebelum pengiriman data kedua, dengan tujuan menyesuaikan alur mekanik proses sortir serta memastikan kestabilan keputusan deteksi sebelum perintah dikirimkan.

Hasil pengujian menunjukkan bahwa Waktu Kirim Serial pada pengiriman pertama dan kedua relatif kecil dan konsisten, masing-masing sebesar 14,96 ms dan 14,78 ms. Nilai ini menunjukkan bahwa proses transmisi data melalui protokol UART berlangsung cepat dan stabil. Selisih waktu kirim serial yang sangat kecil antar pengiriman mengindikasikan bahwa komunikasi UART antara Raspberry Pi 5 dan STM32 memiliki efisiensi tinggi serta tidak menjadi bottleneck utama dalam sistem.

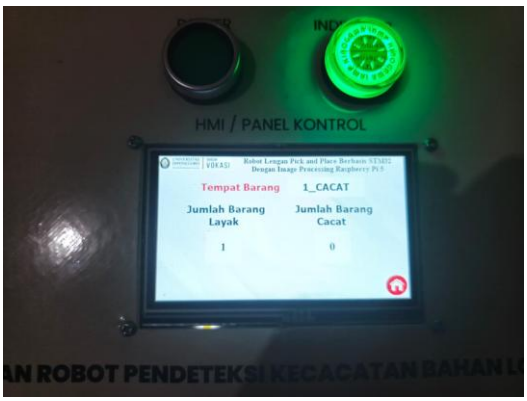
Sebaliknya, Latensi Total Sistem menunjukkan nilai yang jauh lebih besar, yaitu 5018,49 ms pada pengiriman pertama dan 20039,43 ms pada pengiriman kedua. Besarnya nilai latency ini sejalan dengan implementasi delay sistem yang telah dirancang sebelumnya. Pada pengiriman pertama, latensi total mendekati 5000 ms, yang merepresentasikan delay tunggu 5 detik sebelum data dikirim, ditambah dengan waktu pemrosesan deteksi dan waktu kirim serial. Demikian pula pada pengiriman kedua, latensi total mendekati 20000 ms, yang berasal dari delay tunggu 15 detik, akumulasi waktu proses sistem, serta waktu transmisi UART.

Dari hasil tersebut dapat disimpulkan bahwa komponen delay buatan (*intentional delay*) merupakan faktor dominan yang mempengaruhi latensi total sistem, sedangkan kontribusi dari komunikasi UART sangat kecil dibandingkan total waktu tunda yang terjadi. Dengan demikian, protokol UART terbukti cukup andal dan responsif sebagai backbone komunikasi serial dalam sistem ini. Namun, dari sudut pandang sistem waktu nyata (*real-time*), nilai latensi total yang tinggi menunjukkan bahwa respons sistem lebih dipengaruhi oleh kebutuhan sinkronisasi mekanik dan strategi kontrol alur proses, bukan oleh keterbatasan komunikasi data. Secara keseluruhan, hasil analisis ini menunjukkan bahwa sistem telah bekerja sesuai dengan rancangan, di mana latency yang terjadi bersifat terkontrol dan disengaja.

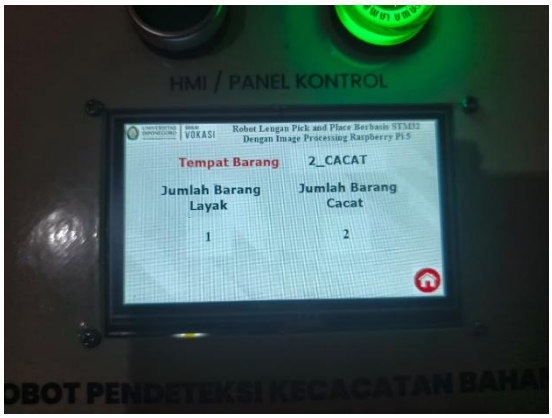
4.6.2. Uji Kirim *Output* Data Barang Cacat

Pengujian uji kirim output data barang cacat dilakukan dengan menempatkan produk yang terdeteksi cacat pada posisi 1 dan posisi 2 secara bergantian pada sistem sortir. Tujuan dari pengujian ini adalah untuk memastikan bahwa hasil deteksi kecacatan yang dilakukan oleh sistem dapat dikirim dan ditampilkan dengan

benar pada antarmuka HMI Nextion. Berdasarkan salah satu hasil pengujian (pengujian 2), ketika barang cacat berada pada posisi 1, HMI menampilkan keluaran berupa teks “1_CACAT”, sedangkan ketika barang cacat juga berada pada posisi 2, HMI menampilkan keluaran “2_CACAT”. Tampilan hasil tersebut menunjukkan bahwa sistem mampu mengidentifikasi posisi barang cacat secara akurat serta mengirimkan informasi status kecacatan sesuai dengan posisi yang terdeteksi. Hasil pengujian ini dapat dilihat pada **Gambar 4.21** dan **Gambar 4.22**. yang memperlihatkan bahwa komunikasi data antara Raspberry Pi 5 dan STM32F411CEU6 yang ditampilkan di HMI Nextion berjalan dengan baik dan sesuai dengan perancangan yang telah dilakukan.



Gambar 4.21 Tempat Barang Cacat Pada Posisi 1



Gambar 4.22 Tempat Barang Cacat Pada Posisi 2

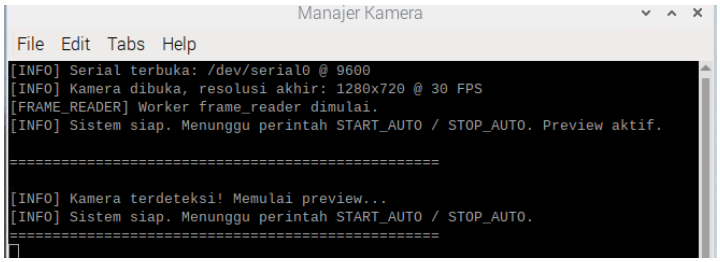
4.7.Pengujian dan Analisa Uji Fungsional Program Auto Run

Pengujian ini mengacu pada file direktori yang dibuat berdasarkan sub bab 3.6.6. yakni *~/.config/autostart/manajer_proses.desktop*. File tersebut merupakan

sebuah entri *desktop* yang digunakan oleh OS berbasis Linux termasuk OS yang digunakan Raspberry Pi untuk secara otomatis menjalankan suatu program saat pengguna *login*. Pengujian dibedakan menjadi 3 yakni sistem ketika dihidupkan, kirim perintah serial *START_AUTO*, dan kirim perintah serial *STOP_AUTO.*

a. Sistem dihidupkan (tampilan awal)

   Program dapat berjalan saat pertama kali *booting* dengan indikasi terdapat LXTerminal yang muncul. Hal ini dapat terjadi karena konfigurasi file manajer_proses.desktop telah ditempatkan dengan benar di direktori ~/.config/autostart/ dan lingkungan desktop (pada Raspberry Pi) berhasil membacanya dan menjalankan perintah Exec saat *startup*. Berikut **Gambar 4.23** dibawah ini merupakan sistem ketika *startup*.



Gambar 4.23 Program saat pertama kali *booting*

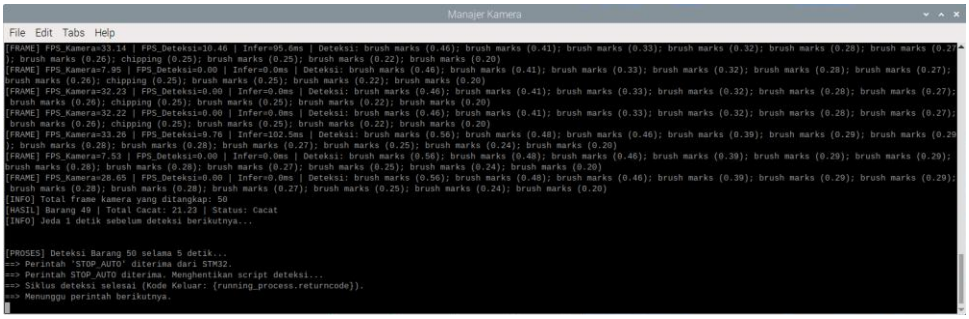
b. Menerima perintah serial *START_AUTO* (dari STM32)

   Modul pembaca serial (di dalam manajer_proses.py) berfungsi dengan baik untuk menerima dan menginterpretasikan perintah dari STM32 sehingga program dapat menerima perintah serial *START_AUTO* yang kemudian dilanjutkan dengan pembacaan program selanjutnya atas dasar perintah dari manajer_proses.py untuk memicu eksekusi tugas utama yaitu *script* deteksi. Berikut **Gambar 4.24** dibawah ini merupakan tampilan LXTerminal ketika menerima perintah serial *START_AUTO*.

Gambar 4.24 Sistem ketika menerima perintah serial *START_AUTO*

c. Menerima perintah serial *STOP_AUTO* saat deteksi aktif

Program merespons interupsi kontrol yakni *STOP_AUTO* secara *real-time* yang kemudian program melakukan penghentian *script* deteksi dan menandakan bahwa siklus deteksi telah selesai. Tak hanya itu, program siap menerima perintah *START_AUTO* kembali tanpa harus melakukan *restart* sistem secara manual. Berikut **Gambar 4.25** dibawah ini merupakan tampilan LXTerminal ketika sistem menerima perintah *STOP_AUTO* dari STM32.



Gambar 4.25 Sistem ketika menerima perintah *STOP_AUTO*

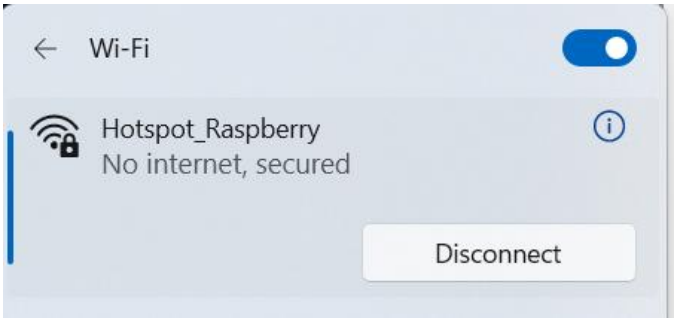
Pengujian fungsi Auto Run melalui manajer_proses.desktop di direktori ~/.config/autostart/ menunjukkan bahwa program berjalan sesuai rencana. Ada tiga poin utama yang berhasil dilakukan. Pertama, program otomatis berjalan saat *booting*, ditandai dengan LXTerminal yang langsung membuka dan menjalankan skrip Manajer Proses. Kedua, program mampu menerima perintah *START_AUTO* dari STM32 lewat komunikasi serial dan dapat langsung memulai proses deteksi. Terakhir, perintah *STOP_AUTO* juga berhasil dihentikan secara real-time saat deteksi berlangsung, tanpa perlu restart aplikasi. Secara keseluruhan, fitur Auto Run dapat bekerja dan sepenuhnya bisa dikendalikan dari luar melalui komunikasi serial.

4.8. Pengujian Konektivitas Access Point dan Internet Sharing

4.8.1. Deteksi dan Koneksi Access Point

Pengujian ini dilakukan untuk memastikan bahwa Raspberry Pi 5 berhasil berfungsi sebagai access point dan dapat menyediakan koneksi jaringan nirkabel bagi perangkat klien. Hasil pengujian menunjukkan bahwa SSID “Hotspot_Raspberry” berhasil dipancarkan dan terdeteksi pada perangkat laptop

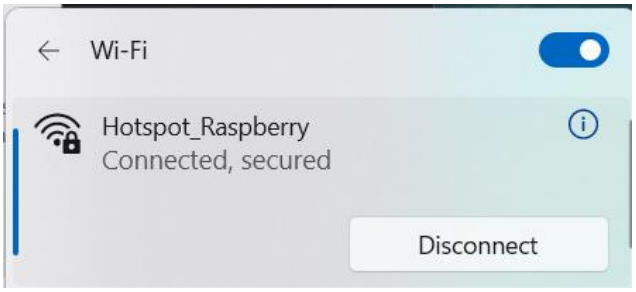
pengguna. Laptop dapat terhubung ke hotspot dengan status koneksi secured serta memperoleh alamat IP secara otomatis dari Raspberry Pi. Namun, pada tahap ini koneksi masih menunjukkan status “No Internet, Secured” dapat dilihat pada **Gambar 4.26**, yang menandakan bahwa access point hanya berfungsi sebagai jaringan lokal dan belum menyediakan akses ke jaringan internet.



Gambar 4.26 *Hotspot_Raspberry* berstatus “No Internet, Secured”

4.8.2. Aktivasi Internet Sharing

Pengujian aktivasi internet sharing dilakukan untuk mengevaluasi keberhasilan konfigurasi IP forwarding dan penerapan NAT menggunakan iptables pada Raspberry Pi. Setelah fitur internet sharing diaktifkan, laptop yang terhubung ke “Hotspot_Raspberry” berhasil memperoleh akses internet, yang ditandai dengan perubahan status koneksi menjadi “Connected, Secured” serta kemampuan perangkat klien untuk mengakses jaringan eksternal dapat dilihat pada **Gambar 4.27**.



Gambar 4.27 *Hotspot_Raspberry* berstatus “Connected, Secured”

Pengujian ini juga menunjukkan bahwa Raspberry Pi 5 memerlukan antarmuka jaringan tambahan berupa dongle Wi-Fi sebagai sumber koneksi internet, sementara antarmuka lainnya digunakan sebagai access point. Selain itu, kelancaran fungsi internet sharing sangat dipengaruhi oleh kesesuaian band

frekuensi Wi-Fi yang didukung oleh dongle. Apabila dongle Wi-Fi hanya mendukung band 2,4 GHz, maka jaringan Wi-Fi yang memaksa berjalan pada band 5 GHz tidak akan terdeteksi dan tidak dapat tersambung ke Raspberry Pi. Oleh karena itu, kesesuaian band frekuensi antara sumber internet dan perangkat dongle menjadi faktor penting dalam keberhasilan koneksi internet sharing.

4.8.3. Otomatisasi Setelah Reboot

Pengujian ini bertujuan untuk memastikan bahwa konfigurasi access point dan internet sharing tetap berjalan secara otomatis setelah Raspberry Pi mengalami proses reboot. Berdasarkan hasil pengujian, hotspot “Hotspot_Raspberry” dapat aktif kembali secara otomatis tanpa memerlukan konfigurasi manual, dan perangkat klien dapat langsung terhubung seperti pada kondisi normal. Selain itu, fungsi internet sharing juga tetap berjalan dengan baik setelah reboot, sehingga perangkat klien tetap memperoleh akses internet secara stabil. Hasil pengujian ini menunjukkan bahwa mekanisme otomatisasi menggunakan service systemd serta penyimpanan aturan iptables telah berjalan dengan baik dan mendukung keandalan sistem dalam penggunaan berkelanjutan.