**IJEIS (Indonesian Journal of Electronics and Instrumenations Systems)**
Vol.15, No.1, April 2025, pp. 49~60
ISSN (print): 2088-3714, ISSN (online): 2460-7681
DOI: 10.22146/ijeis.104520


# Sistem Klasifikasi Sampah Otomatis Berbasis Deteksi Objek *Real-Time* Pada *Single Board Computer* Dengan Algoritma YOLO

**Ahmad Zaki Firdaus*[1], Danang Lelono[2], Oskar Natan[3]**
[1,2,3]Departemen Ilmu Komputer dan Elektronika, FMIPA UGM, Yogyakarta, Indonesia
e-mail: ***[1]ahmad.zaki.firdaus@mail.ugm.ac.id**, [2]danang@ugm.ac.id, [3] oskarnatan@ugm.ac.id

***Abstrak***

*Pengembangan sistem klasifikasi sampah otomatis berbasis deteksi objek real-time menggunakan algoritma YOLO (You Only Look Once) pada Single Board Computer (SBC) Raspberry Pi 5 menjadi fokus utama dalam tugas akhir ini. Permasalahan utama yang diangkat adalah penumpukan sampah yang semakin meningkat, khususnya di Indonesia, yang memerlukan solusi efektif untuk pemilahan sampah secara otomatis. Sistem ini dirancang untuk mendeteksi dan memilah sampah plastik dan logam secara real-time dengan memanfaatkan teknologi deep learning dan computer vision.*

*Penelitian ini menggunakan model YOLO11n yang dilatih dengan dataset sampah plastik dan logam. Proses pelatihan melibatkan teknik augmentasi data seperti rotasi dan grayscale untuk meningkatkan variasi dataset. Hasil pelatihan menunjukkan akurasi mean Average Precision (mAP) sebesar 98,44% pada data testing. Sistem ini diimplementasikan pada Raspberry Pi 5 dengan konversi model ke format NCNN untuk meningkatkan kecepatan inferensi. Hasil pengujian menunjukkan sistem mampu mencapai kecepatan 8,90 FPS dengan latency 110 ms, yang memenuhi kriteria sistem real-time.*

***Kata kunci****— Deteksi Objek, YOLO11, Single Board Computer, Raspberry Pi*

***Abstract***

*The development of an automatic waste classification system based on real-time object detection using the YOLO (You Only Look Once) algorithm on a Raspberry Pi 5 Single Board Computer (SBC) is the main focus of this final project. The main issue addressed is the increasing accumulation of waste, particularly in Indonesia, which requires an effective solution for automatic waste sorting. The system is designed to detect and sort plastic and metal waste in real-time using deep learning and computer vision technologies.*

*This research employs the YOLO11n model, trained on a dataset of plastic and metal waste. The training process involves data augmentation techniques such as rotation and grayscale to enhance dataset variability. The training results show a mean Average Precision (mAP) of 98.44% on testing data. The system is implemented on a Raspberry Pi 5, with the model converted to NCNN format to improve inference speed. Testing results indicate that the system can achieve a speed of 8.90 FPS with a latency of 110 ms, meeting the criteria for a real-time system.*

***Keywords****—Object Detection, YOLO11, Single Board Computer, Raspberry Pi*

*Received February 9th,2025; Revised April 11th, 2025; Accepted April 29th, 2025*

## 1. PENDAHULUAN

Permasalahan sampah global semakin mengkhawatirkan. Menurut World Bank (2023), produksi sampah global mencapai 2,01 miliar ton pada tahun 2020 dan diproyeksikan meningkat menjadi 3,4 miliar ton pada 2050 [1]. Pemerintah Indonesia telah mengumumkan target untuk mengurangi sampah hingga 30% pada tahun 2025 [2]. Di Indonesia, Kementerian Lingkungan Hidup dan Kehutanan (KLHK) melaporkan bahwa pada tahun 2022, total sampah yang dihasilkan mencapai 18,08 juta ton per tahun, dengan hanya 65,92% yang terkelola. Target pemerintah adalah mengurangi 30% sampah pada tahun 2025 melalui program pengelolaan sampah berbasis teknologi [3]. Jumlah sampah yang belum terkelola dengan baik bukanlah jumlah yang sedikit.

Untuk mengatasi permasalahan sampah yang menumpuk, langkah penting untuk mencapai target pengurangan sampah adalah melakukan pemilahan sampah seawal mungkin. Hal ini dilakukan agar sampah yang menumpuk tidak saling terkontaminasi satu sama lain. Terkontaminasinya sampah ini dapat membahayakan lingkungan sekitarnya dan membahayakan keselamatan pekerja yang bertugas untuk memilah sampah tersebut jika sampah mengandung bahan berbahaya.

Dengan melakukan pemilahan sampah sebelum dibuang, hal ini dapat menjadi langkah yang efektif karena sampah yang dibuang sudah terpisah sesuai dengan jenisnya sehingga tidak perlu mengeluarkan tenaga untuk memilah. Tempat sampah yang dipisah sesuai dengan jenisnya pada kawasan kecil seperti fasilitas publik dapat menjadi langkah awal untuk memulai kebiasaan baik memilah sampah sebelum dibuang. Namun, tak jarang ditemukan sampah masih bercampur bahkan tidak dibuang pada tempatnya. Hal ini dapat terjadi karena rendahnya kepedulian masyarakat terhadap pengelolaan sampah. Berdasarkan survei yang dilakukan oleh BPS, indeks ketidakpedulian masyarakat terhadap pengelolaan sampah di Indonesia sebesar 72%. Indeks tersebut menandakan bahwa hanya sedikit masyarakat yang peduli terhadap pengelolaan sampah [4].

Deteksi objek merupakan salah satu teknik dalam *computer vision* untuk menemukan dan mengidentifikasi sebuah objek yang terdapat dalam bentuk gambar ataupun video. Pada beberapa tahun terakhir deteksi objek menjadi topik penting dalam *computer vision*. Implementasi deteksi objek antara lain seperti pada *face recognition*, *anomaly detection*, *quality control* dan *Autonomous Vehicle*.

Teknik deteksi objek bekerja dengan cara melakukan lokalisasi objek pada pengolahan citra dengan memberi kotak batasan atau *bounding box* yang secara bersamaan melakukan klasifikasi setiap objek di dalam sebuah gambar ataupun video [5]. Deteksi objek adalah gabungan dari lokalisasi objek dan klasifikasi objek sehingga dapat mencari letak objek yang kita inginkan pada suatu gambar serta mengklasifikasikannya gambar tersebut sesuai dengan label yang dibuat.

Untuk melakukan deteksi objek diperlukan algoritma. YOLO (You Only Look Once) merupakan salah satu algoritma deep learning untuk melakukan deteksi objek, yang merupakan peningkatan lebih lanjut atau versi terbaru dari seri YOLO sebelumnya. Algoritma ini memiliki performa *real-time* dan juga akurasi deteksi yang tinggi dan struktur jaringan yang ringan [6], cocok untuk dilakukan deteksi objek pemilahan jenis sampah otomatis secara cepat dan akurat.

Beberapa studi telah dilakukan untuk melakukan pemilahan sampah menggunakan deteksi objek. Salah satunya menggunakan YOLOv4 yang dimodifikasi sebagai modelnya yang menggunakan dataset dari TrashX & TrashNet Dataset dan mengaplikasikannya kepada *mobile device* Apple iPhone [7]. Selain itu ada juga studi lain yang mengimplementasikannya kepada robot humanoid untuk memisahkan berbagai sampah tipe plastik [8]. Studi lain yang berkaitan kebanyakan memanfaatkan model *one-stage object detection* yang menggabung proses deteksi objeknya seperti YOLO dan versi-versi lainnya dan juga meningkatkan model YOLO.

Dari beberapa studi yang telah dilakukan untuk mendeteksi objek pada pemilahan sampah, kebanyakan mengaplikasikannya hanya dengan inputan gambar [9] dan penangkapan gambar melalui kamera mobile [10]. Aplikasi sistem deteksi objek tersebut tentunya membutuhkan perangkat yang portable dan mendukung edge computing, dimana *edge computing*

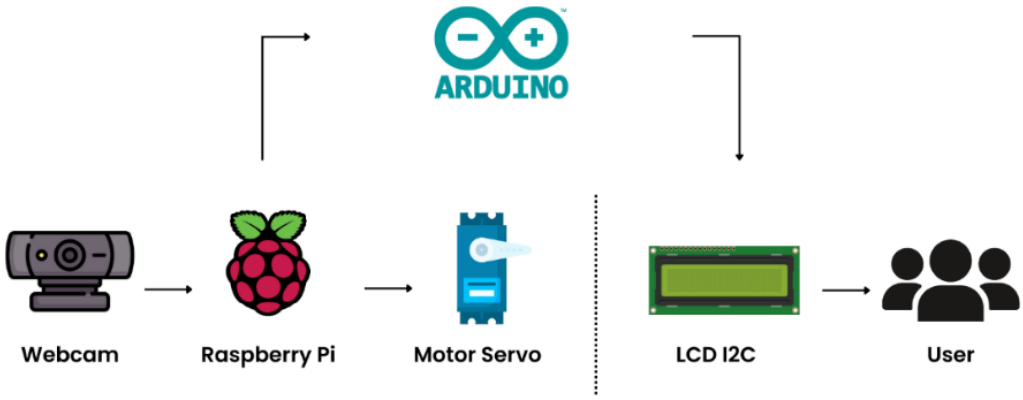
sendiri adalah sebuah sistem yang melakukan komputasi berada pada atau di dekat sumber data [11]. Sehingga tidak harus membutuhkan koneksi internet untuk menjalankan model pada *cloud* dan dapat meningkatkan mobilitas. Beberapa contoh perangkat portable yang mendukung *edge computing* diantaranya adalah *single board computer* seperti NVIDIA Jetson Series, Raspberry Pi, Orange Pi, Intel Movidius dan Google coral.

Dari beberapa informasi yang sudah ada sebelumnya, maka latar belakang dari penelitian ini adalah agar dapat mengembangkan sebuah sistem pemilahan sampah dengan memanfaatkan deteksi objek berbasis *Deep Learning* dengan metode deteksi objek yang efisien serta *robust*. Dalam hal ini penulis mengusulkan penggunaan metode terbaru YOLO11 untuk proses deteksi objek sampah secara *real-time* sehingga diharapkan dapat diimplementasikan pada *single board computer*.

## 2. METODE PENELITIAN

### *2. 1 Rancangan Perangkat Keras*

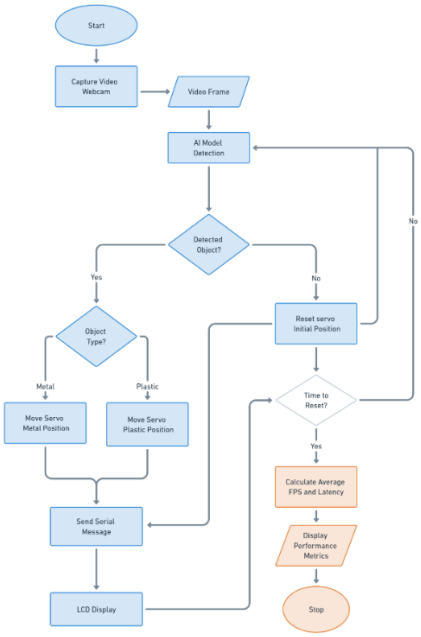
Sistem ini terdiri dari beberapa komponen utama yang memiliki peran spesifik dalam operasinya. Webcam berfungsi sebagai perangkat input untuk menangkap citra sampah secara real-time. Raspberry Pi digunakan sebagai *single board computer (SBC)* yang bertanggung jawab untuk menjalankan program utama dan memproses data. Motor Servo diimplementasikan sebagai aktuator yang menggerakkan mekanisme pemilah sampah berdasarkan hasil deteksi objek. Arduino Nano digunakan sebagai komunikasi data antara Raspberry Pi. LCD I2C berperan sebagai antarmuka pengguna untuk menampilkan informasi status sistem melalui Arduino Nano. Rancangan desain perangkat keras sistem dapat dilihat pada **Gambar 1** berikut.




Gambar 1 Rancangan Perangkat Keras

### *2. 2 Rancangan Perangkat Lunak*

Sistem ini memanfaatkan kamera webcam untuk menangkap video, yang kemudian diproses oleh *single board computer* untuk diproses menggunakan model YOLO. Model deteksi ini berfungsi untuk mengidentifikasi keberadaan objek dan menentukan jenis objek tersebut, apakah merupakan metal atau plastik. Informasi hasil deteksi ini digunakan untuk menggerakkan motor servo yang berperan sebagai pemilah sampah, memastikan setiap jenis sampah diarahkan ke tempat yang sesuai. Rancangan program sistem dapat dilihat pada **Gambar 2** dibawah ini.



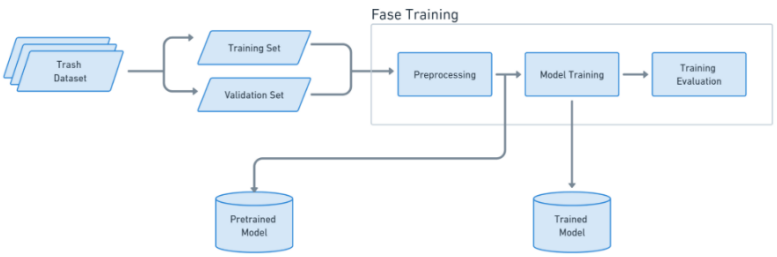

Gambar 2 Alur Program

Untuk memastikan performa optimal, komunikasi serial antara Raspberry Pi dan Arduino digunakan untuk menampilkan informasi proses pada layar LCD. Implementasi LCD pada Arduino dipilih untuk menghindari penurunan performa FPS pada video deteksi, yang dapat terjadi jika LCD diintegrasikan langsung ke Raspberry Pi. Informasi yang ditampilkan meliputi indikator objek yang terdeteksi dan status sistem.

*2. 3 Rancangan Model Deteksi*

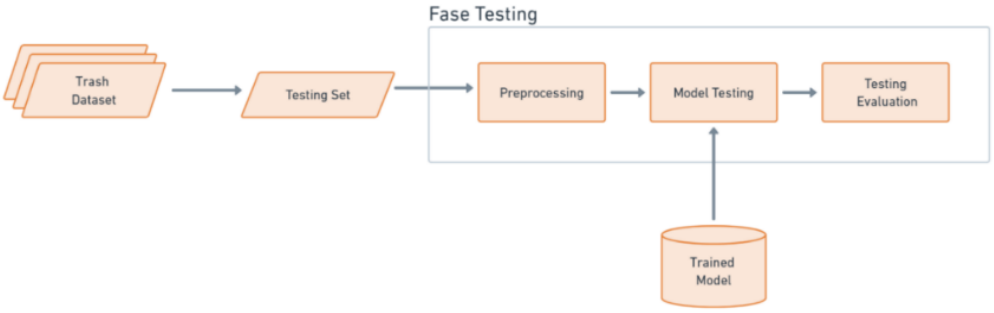
Pada tahap ini akan digunakan model YOLO11 sebagai model deteksi objek. Proses penelitian mencakup dua fase utama, yaitu *training* dan *testing*. Dataset gambar diperlukan untuk melakukan dua fase ini yang akan membantu model untuk memahami dan mendeteksi objek tertentu berdasarkan data yang diberikan. Dataset yang terkumpul akan dibagi menjadi 3 jenis set yaitu *training set, validation set*, dan *testing set* dengan perbandingan 80%, 15%, dan 5%.

Tahap pelatihan akan menggunakan *training set* dan *validation set*. *Training set* digunakan untuk melatih model, sedangkan *validation set* digunakan untuk mendapatkan validasi pelatihan dan mencegah model mengalami *overfitting* saat pelatihan berlangsung. Diagram fase *training* dapat dilihat pada **Gambar 3** di bawah ini.




Gambar 3 Fase *Training*

Fase pengujian akan mengevaluasi model pendeteksian dengan menggunakan dataset yang belum pernah dilihat oleh model sebelumnya saat pelatihan yang merupakan *testing set*. Diagram fase *testing* dapat dilihat pada **Gambar 4** di bawah ini.




Gambar 4 Fase *Testing*

*2. 4 Pengujian dan Evaluasi Sistem*

Pengujian dan evaluasi sistem dilakukan dengan mengukur tingkat keberhasilan algoritma deteksi objek yang telah dibuat. Rencana pengujian sistem meliputi evaluasi kinerja model yang dapat dilihat pada **Tabel 1** di bawah ini.

Tabel 1 Rencana Pengujian Sistem

| No | Rencana Pengujian | Tujuan Pengujian |
|---|---|---|
| 1 | Performa Model Deteksi | Mendapatkan performa akurasi model terbaik pada sistem. |
| 2 | Performa Komputasi Deteksi | Untuk memeriksa kinerja sistem berdasarkan *latency*-nya |

Proses pengujian sistem ini dilakukan untuk menguji apakah model mampu membedakan jenis sampah berdasarkan kategorinya yaitu sampah plastik dan sampah metal dengan benar menggunakan data uji (*testing*) yang sudah disiapkan dan menggunakan citra yang belum pernah digunakan saat proses pembuatan model. Pengujian performa sistem akan dilakukan dengan memanfaatkan *confusion matrix* yang dihitung dari rekaman hasil deteksi secara *real-time* dan hasil perhitungan *latency* yang telah diterapkan dalam program. Untuk keterangan lebih lanjut, dapat dilihat pada **Tabel 2** sebagai berikut.

Tabel 2 *Performance Metrics* untuk model deteksi

| No | Performance Metrics | Fungsi |
|---|---|---|
| 1 | *Precision* | Mengetahui persentase dalam mengenali objek deteksi positif. |
| 2 | *Recall* | Mengetahui persentase dalam mengenali objek yang relevan. |
| 3 | *F1 Score* | Mengetahui akurasi hasil pengujian. |
| 4 | *Average Precision (AP)* | Mengetahui performa model di semua *threshold*. |
| 5 | *mean Average Precision (mAP)* | Mengetahui rata-rata dari nilai *Average Precision*. |

Performa komputasi deteksi digunakan untuk mengukur kecepatan sistem apakah sudah cukup cepat untuk diklasifikasikan sebagai sistem real-time. Performance metrics untuk performa kecepatan sistem dapat dilihat pada **Tabel 3** dibawah ini.

Tabel 3 *Performance Metrics* untuk komputasi deteksi

| No | Performance Metrics | Fungsi |
|---|---|---|
| 1 | Latensi | Mengetahui waktu respon sistem saat mengenali objek. |
| 2 | FPS | Mengetahui seberapa cepat sistem berjalan saat melakukan deteksi objek. |

Persamaan untuk menghitung semua metrik kinerja tersebut dapat dilihat di bawah ini.

$$IoU = \frac{Area\ of\ Intersection}{Area\ of\ Union} \tag{1}$$

$$Precision = \frac{True\ Positive}{Actual\ Results} = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{True\ Positive}{Predicted\ Results} = \frac{TP}{TP + FN} \tag{3}$$

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

$$mAP@\alpha = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{5}$$

$$Latency = t_{frame} - t_{before} \tag{6}$$

$$FPS = \frac{1}{Latency} \tag{7}$$

Pengujian yang dilakukan diharapkan mencapai hasil dengan parameter yang terdapat pada **Tabel 4** berikut.

Tabel 4 Parameter Indikator Penelitian

| No | Parameter | Indikator Keberhasilan |
|---|---|---|
| 1 | Akurasi Deteksi | Dapat meraih mAP sebesar 90% atau lebih. |
| 2 | Komputasi Deteksi | Dapat meraih rata-rata latensi per deteksi kurang dari 150 ms dengan konsumsi sumber data (CPU, RAM) stabil pada SBC. |

Akurasi deteksi ini penting deteksi objek, agar tidak terjadi kesalahan dalam mendeteksi suatu objek, sedangkan latensi deteksi mempengaruhi kecepatan untuk sistem merespon dari hasil deteksi objek untuk sistem mencapai kategori *real-time*.

## 3. HASIL DAN PEMBAHASAN

### *3. 1 Analisis Dataset*

Dataset yang digunakan dalam penelitian ini terdiri dari 1250 gambar dengan distribusi kelas metal berjumlah 569 data dan plastic 681 data. Sebelum pelatihan model, seluruh gambar diresize ke resolusi 640×640 piksel dan dinormalisasi ke rentang [0, 1] untuk memenuhi kebutuhan input arsitektur YOLO11. Pembagian dataset dilakukan dalam dua skenario pada percobaan pertama menggunakan rasio 80-10-10 (1000 gambar latih, 125 validasi, 125 uji), sedangkan percobaan kedua mengadopsi rasio 80-15-5 (1000 latih, 190 validasi, 60 uji) untuk mengevaluasi pengaruh ukuran subset validasi terhadap generalisasi model.

Augmentasi data diterapkan secara dinamis selama pelatihan guna meningkatkan variasi dataset yang dapat dilihat pada **Tabel 5**. Pada eksperimen pertama, teknik augmentasi meliputi *flip horizontal/vertical*, rotasi 90°, *crop* 0-20%, rotasi ±15°, dan *shear* ±15°. Sementara itu, eksperimen kedua menerapkan augmentasi lebih intensif dengan rotasi ±45° dan penambahan *grayscale* pada 20% gambar untuk mensimulasikan kondisi pencahayaan yang beragam.

Tabel 5 Komposisi dan Karakteristik Dataset

| Parameter | Eksperimen 1 | Eksperimen 2 |
|---|---|---|
| Total Gambar | 1250 | 1250 |
| Split Dataset | Train: 1000<br>Valid: 125<br>Test: 125 | Train: 1000<br>Valid: 190<br>Test: 60 |
| Resolusi Gambar | 640 x 640 piksel | 640 x 640 piksel |
| Augmentasi | Flip horizontal/vertikal<br>Rotasi 90°<br>Crop 0-20%<br>Rotasi ±15°<br>Shear ±15° | Flip horizontal/vertikal<br>Rotasi 90°<br>Crop 0-20%<br>Rotasi ±45°<br>Shear ±15°<br>Grayscale 20% |

### *3. 2 Eksperimen Hyperparameter Tuning*

*Hyperparameter* yang akan dibahas adalah tentang *Batch Size*. Variasi atau pengaturan dari *batch size* tidak selalu mempengaruhi akurasi dari suatu model, meskipun beberapa kali berpengaruh yang disebabkan oleh *randomness* inisialisasi bobot pada awal pelatihan [12]. Tetapi yang paling berpengaruh dari *batch size* adalah kebutuhan komputasinya yang semakin tinggi seiring meningkatnya *batch size*, dikarenakan model akan lebih banyak menyimpan memori pada RAM terutama jika dilatih pada GPU. Sedangkan seiring menurunnya jumlah *batch size* akan berpengaruh juga dengan waktu latih model meskipun dengan jumlah *epochs* yang sama yaitu 100 *epochs*, dikarenakan proses pelatihan akan lebih banyak melakukan perhitungan *loss* dan pembaruan bobot model yang dibuktikan dengan rumus perhitungan iterasi pada persamaan 8.

$$Iterations = \frac{datasize}{batchsize} \quad (8)$$

Dalam eksperimen ini, difokuskan pada pengujian batch size 16, 24, 32, dan 64 terhadap dua skenario dataset seperti yang ada pada Tabel 6.1. Untuk batas atas ditentukan 64 karena jika melebihi itu akan terjadi OOM atau *Out of Memory* pada GPU sehingga pelatihan tidak dapat dilanjutkan. Sedangkan pada batas bawah ditentukan 16 karena waktu latihnya yang agak lama dan sudah cukup mempresentasikan sebagai ukuran yang lebih kecil.

Pada eksperimen pertama, seperti terlihat pada **Tabel 6**, *batch size* 32 menghasilkan mAP tertinggi yaitu 97,85% dengan waktu pelatihan 1,265 jam/epoch. Namun, peningkatan *batch size* menjadi 64 menyebabkan penurunan pada metrik *recall* hingga 92,04%, mengindikasikan risiko *underfitting*.

Tabel 6 Hasil Pelatihan pada Dataset 1 (Split 80-10-10)

| Batch | Precision | Recall | F1- Score | AP | | mAP | Training Time | Inference Time |
|---|---|---|---|---|---|---|---|---|
| | | | | Metal | Plastic | | | |
| 16 | 93,76% | 95,20% | 94,67% | 96,35% | 96,37% | 96,36% | 1,365 jam | 3,3 ms |
| 24 | 94,66% | 94,47% | 94,85% | 95,08% | 97,17% | 96,13% | 1,182 jam | 2,5 ms |
| 32 | 92,83% | 96,81% | 95,40% | 97,49% | 98,21% | 97,85% | 1,265 jam | 2,4 ms |
| 64 | 93,99% | 92,04% | 93,68% | 96,75% | 96,22% | 96,48% | 1,272 jam | 2,3 ms |

Sementara itu pada eksperimen kedua, seperti terlihat pada **Tabel 7** menunjukkan bahwa *batch size* 24 mAP mencapai 99,12%, tetapi waktu pelatihan lebih lama, karena kompleksitas augmentasi rotasi ±45° dan *grayscale*. Pada perbedaan waktu latihnya lumayan signifikan terutama pada *batch size* 64, 32, 24, lalu pada *batch size* 16 menyentuh waktu 2,029 jam untuk waktu pelatihan, oleh karena itu penulis tetap memilih model dengan *batch size* 24 pada eksperimen kedua untuk digunakan pada *deployment* dan eksperimen berikutnya.

Tabel 7 Hasil Pelatihan pada Dataset 2 (Split 80-15-5)

| Batch | Precision | Recall | F1- Score | AP | | mAP | Training Time | Inference Time |
|---|---|---|---|---|---|---|---|---|
| | | | | Metal | Plastic | | | |
| 16 | 96,09% | 96,42% | 96,21% | 97,80% | 98,24% | 98,02% | 2,029 jam | 4,0 ms |
| 24 | 96,84% | 97,84% | 98,26% | 98,89% | 99,35% | 99,12% | 1,665 jam | 2,5 ms |
| 32 | 95,19% | 96,05% | 95,31% | 98,40% | 98,32% | 98,36% | 1,617 jam | 2,5 ms |
| 64 | 96,05% | 98,14% | 97,68% | 98,28% | 98,50% | 98,39% | 1,563 jam | 2,4 ms |

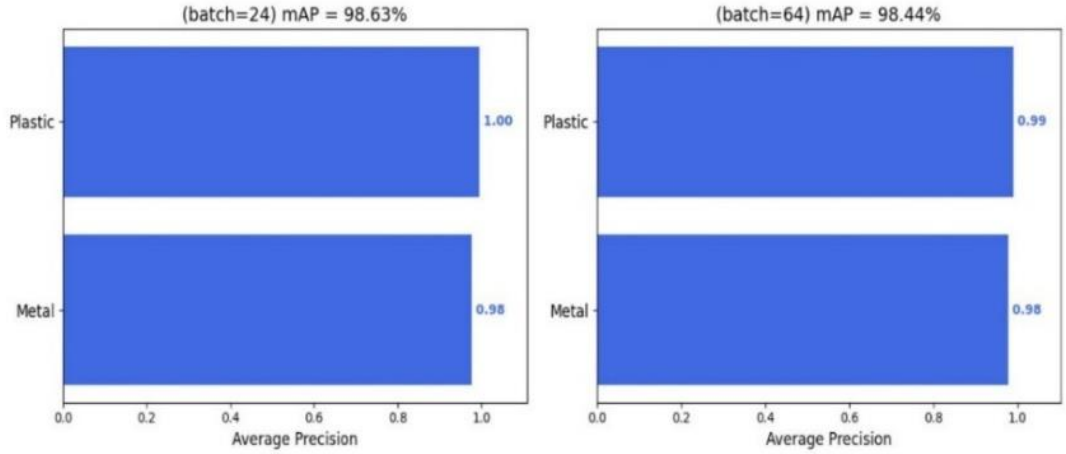
*3. 3 Analisis Pelatihan Model YOLO11*

Berdasarkan hasil eksperimen, *batch size* 24 dan 64 pada eksperimen kedua dipilih sebagai konfigurasi yang paling optimal antara kedua eksperimen dataset. *Batch size* 24 mendapatkan akurasi yang lebih tinggi diantara yang lain, sedangkan pada *batch size* 64 memiliki *inference time* yang lebih ringan.3. 3 Hasil Pengujian Model YOLO11

*3. 4 Hasil Pengujian Data Testing*

Pengujian yang dilakukan pada penelitian ini adalah menguji partisi data testing yang sudah dibagi sebelumnya sebanyak 60 data. Tujuan dari pengujian ini adalah melihat apakah

model mengalami *overfit* atau *underfit* dan tetap memiliki akurasi setinggi hasil pelatihan meskipun diuji dengan data yang belum pernah dilihat sebelumnya yaitu data testing.




Gambar 5 Hasil Akurasi mAP dengan Data Testing

Berdasarkan pengujian yang telah dilakukan pada data testing, didapatkan hasil akurasi mAP sebesar 98,63% untuk *batch size* 24 dan mAP sebesar 98,44% untuk *batch size* 64 yang dapat dilihat pada **Gambar 5**. Pada kedua nilai mAP mengindikasikan bahwa model tidak mengalami *overfitting* atau *underfitting*. Stabilitas ini tercapai berkat teknik augmentasi yang diterapkan selama pelatihan, yang meningkatkan variasi data sehingga model dapat beradaptasi dengan kondisi nyata. Grafik ini juga menegaskan bahwa konfigurasi *hyperparameter* yang digunakan termasuk *batch size* telah optimal untuk menjaga generalisasi model.

Tabel 8 Perbedaan Akurasi Model pada Data Testing dan Data Training

| **Batch Size** | **Data** | | **Selisih** | **Batch Size** | **Data** | | **Selisih** |
|---|---|---|---|---|---|---|---|
| **24** | **Train** | **Test** | | **64** | **Train** | **Test** | |
| Plastic AP | 99,35% | 99,52% | 0,17% | Plastic AP | 98,50% | 99,05% | 0,55% |
| Metal AP | 98,89% | 97,73% | -1,16% | Metal AP | 98,28% | 97,83% | -0,45% |
| mAP@0.5 | 99,12% | 98,63% | -0,49% | mAP@0.5 | 98,39% | 98,44% | 0,45% |
| Precision | 96,84% | 95,56% | -1,28% | Precision | 96,05% | 96,67% | 0,62% |
| Recall | 97,84% | 98,89% | 1,05% | Recall | 98,14% | 98,91% | 0,77% |
| F1-Score | 98,26% | 97,19% | -1,07% | F1-Score | 97,68% | 97,78% | 0,10% |

Selain itu pada **Tabel 8** ditunjukkan juga log dari metrik lainnya dimana pada *batch size* 24, mAP@0.5 secara keseluruhan mengalami penurunan 0,49%. Penurunan *precision* sebesar 1,28% dan peningkatan *recall* sebesar 1,05% mengindikasikan bahwa model menjadi lebih sensitif dalam mendeteksi objek tetapi kurang presisi (*false positive* meningkat).

Di sisi lain, *batch size* 64 menunjukkan konsistensi lebih baik, mAP@0.5 secara keseluruhan naik 0,45%. Peningkatan *precision* 0,62% dan *recall* 0,77% pada data testing menunjukkan bahwa model dengan *batch size* 64 tidak hanya akurat tetapi juga stabil dalam mendeteksi kedua kelas. Selisih *F1-Score* yang kecil +0,10% menegaskan keseimbangan antara *precision* dan *recall*. Hasil ini membuktikan bahwa augmentasi intensif dan split dataset 80-15-5 berhasil mengurangi *overfitting*, bahkan pada *batch size* besar.

### *3. 5 Analisis Pengujian Model YOLO11*

Berdasarkan hasil pengujian yang telah dilakukan terhadap *hyperparameter*, penulis memilih *batch size* 64 sebagai solusi optimal untuk *deployment* pada Raspberry Pi 5. Meskipun *batch size* 24 menunjukkan akurasi yang lebih tinggi, pertimbangan utama penelitian ini adalah

menyeimbangkan kecepatan inferensi dan efisiensi komputasi tanpa mengorbankan performa deteksi secara signifikan.

*Batch size* 64 memberikan keunggulan dalam hal stabilitas waktu inferensi dan konsumsi sumber daya yang lebih rendah, yang kritis untuk sistem *real-time* pada perangkat *edge* dengan kemampuan terbatas. Pemilihan ini selaras dengan tujuan penelitian untuk mengoptimalkan responsivitas sistem dalam skenario pemilahan sampah aktual, di mana kecepatan deteksi dan keandalan operasional menjadi faktor penentu. Dengan demikian, model ini tidak hanya memenuhi kriteria akurasi minimal yang ditetapkan, tetapi juga menjamin kelancaran implementasi di lingkungan nyata, di mana *latency* rendah dan konsistensi performa menjadi prioritas utama.

### *3. 6 Hasil Pengujian Komputasi Model YOLO11*

Pengujian komputasi pada *Single Board Computer (SBC)* Raspberry Pi 5 bertujuan untuk mengevaluasi kemampuan sistem dalam menjalankan deteksi objek secara *real-time* dengan parameter *latency* dan *frame per second (FPS).* Hasil pengujian dapat dilihat pada **Tabel 9** menunjukkan perbandingan performa model menggunakan dua *framework* berbeda dengan variasi *batch size*.

Tabel 9 Hasil Perbandingan Komputasi Model

| Batch | Model Format | FPS | Latency | Memory Usage |
|---|---|---|---|---|
| 24 | PyTorch | 2,71 FPS | 370 ms | 433 MB |
| 64 | PyTorch | 2,74 FPS | 360 ms | 434 MB |
| 24 | NCNN | 7,59 FPS | 130 ms | 456 MB |
| 64 | NCNN | 8,90 FPS | 110 ms | 461 MB |

Pada *framework* PyTorch, baik *batch size* 24 maupun 64, sistem hanya mencapai 2,7 FPS dengan *latency* 360-370 ms, yang belum memenuhi kriteria *real-time*. Namun, setelah dikonversi ke format NCNN , performa meningkat signifikan dengan FPS 7,59 pada *batch* 24 dan 8,90 FPS pada *batch* 64 dengan *latency* 110-130 ms, meskipun penggunaan memori sedikit meningkat pada 461 MB dibandingkan *batch* 24 yang hanya 456 MB. Kenaikan memori ini masih dalam batas toleransi untuk RaspberryPi 5 RAM 8GB, sehingga tidak mengganggu stabilitas sistem. Peningkatan ini terjadi karena NCNN dioptimalkan khusus untuk perangkat *edge* dengan kompresi model dan penggunaan sumber daya yang efisien. Dengan *latency* 110 ms, sistem ini memenuhi indikator keberhasilan yang ditetapkan pada metode penelitian, sehingga layak diimplementasikan untuk sistem deteksi objek pemilahan sampah. Meskipun FPS belum mencapai target awal pada 20 FPS, nilai 8,90 FPS tetap memadai untuk aplikasi pemilahan sampah yang umumnya melibatkan objek statis atau bergerak lambat.

Pada **Tabel 10** dan **Gambar 5** menunjukkan bahwa model YOLO11n dengan konfigurasi *batch size* 64 mampu mencapai *latency* 110 ms dengan FPS tertinggi 8,90 frame/detik pada resolusi video 720p. Kinerja ini mengalami peningkatan signifikan dibandingkan penelitian terdahulu yang menggunakan YOLO-Green dengan FPS hanya 2,72 pada perangkat serupa [13] dan penelitian lainnya mendapatkan *latency* yang sangat besar yaitu 1-5 detik [14].

Tabel 10 Hasil Perbandingan dengan Penelitian Sebelumnya

| Penelitian | Metode | mAP | FPS | Latency |
|---|---|---|---|---|
| Lin, 2021 | YOLO-GREEN | 78% | 2,72 FPS | None |
| Hermawan et al., 2023 | YOLOv5 | 85% | None | 1000 ms |
| **Firdaus, 2025** | **YOLO11** | **98 %** | **8,90 FPS** | **110 ms** |


Gambar 6 Hasil Pengujian Komputasi Model

## 4. KESIMPULAN

Berdasarkan penelitian yang dilakukan, diperoleh kesimpulan bahwa telah berhasil menciptakan sistem yang efisien dan mampu berjalan secara *real-time* di Raspberry Pi 5. Model YOLO11n menunjukkan performa akurasi tinggi dengan mAP 98,44%, *precision* 96,67%, *recall* 98,91%, dan *F1-score* 97,78% pada data testing. Teknik augmentasi rotasi ±45° dan *grayscale* terbukti meningkatkan ketahanan model terhadap variasi orientasi dan pencahayaan meskipun dataset yang digunakan relatif kecil. Penggunaan *batch size* 64 menghasilkan performa lebih optimal dalam testing, meskipun *batch size* 24 menunjukkan akurasi lebih tinggi pada data training. Sistem ini berhasil menyeimbangkan akurasi mAP 98,44% dan efisiensi komputasi dengan kecepatan 8,90 FPS, *latency* 110 ms, serta penggunaan memori 461 MB. Dibandingkan penelitian sebelumnya, model ini unggul dalam *latency* (110 ms vs. 1000-5000 ms) dan kecepatan deteksi (8,90 FPS vs. 2,72 FPS), sehingga memenuhi kriteria *real-time* dan responsivitas tinggi untuk skenario pemilahan sampah aktual.

## 5. SARAN

Untuk pengembangan selanjutnya, disarankan memperluas kualitas dan kuantitas dataset dengan gambar yang lebih konsisten, mengingat dataset saat ini bersumber dari *open source* dengan variasi kondisi pencahayaan dan resolusi yang tidak seragam. Selain itu, menambah jumlah kelas objek yang dapat dideteksi akan meningkatkan robustness sistem dalam skenario pemilahan sampah yang kompleks. Penerapan pada perangkat *Single Board Computer* berkinerja lebih tinggi, seperti NVIDIA Jetson Series, juga perlu dipertimbangkan untuk meningkatkan kecepatan deteksi (FPS) guna mencapai performa *real-time* yang lebih optimal. Dengan demikian, sistem ini dapat lebih adaptif dan efektif dalam aplikasi praktis di lingkungan nyata.

## DAFTAR PUSTAKA

[1] Kaza, S., Yao, L., Bhada-Tata, P. and Van Woerden, F., 2018. What a waste 2.0: A Global Snapshot of Solid Waste Management to 2050. *World Bank Publications*.

[2] Fiona, F. and Fitri, W., 2023. Efektivitas Hukum Lingkungan Dalam Mengurangi Sampah Plastik Di Lautan Indonesia Pada Era Globalisasi. *Gorontalo Law Review, 6(1), pp.155-164.*

[3] Kementerian Lingkungan Hidup dan Kehutanan Republik Indonesia (2024) *Sistem Informasi Pengelolaan Sampah Nasional (SIPSN)*. Available at: https://sipsn.menlhk.go.id/sipsn/ [Accessed: 1-Maret-2024]

[4] Badan Pusat Statistik (2021) Persentase rumah tangga menurut provinsi dan perlakuan memilah sampah mudah membusuk dan tidak mudah membusuk, 2013-2014, 2021 [Online]. Available at: https://www.bps.go.id/id/ [Accessed: 2-Maret-2024]

[5] Shenai, H., Gala, J., Kekre, K., Chitale, P. and Karani, R., 2022. Combating COVID-19 using object detection techniques for next-generation autonomous systems. In *Cyber-Physical Systems* (pp. 55-73). Academic Press.

[6] Yanfei, P. and Yue, J., 2023, November. Road Crack Detection Algorithm Based on Improved YOLOv8. In *2023 5th International Conference on Artificial Intelligence and Computer Applications (ICAICA)* (pp. 28-32). IEEE.

[7] Lin, W., 2021, December. YOLO-green: A real-time classification and object detection model optimized for waste management. In *2021 IEEE International Conference on Big Data (Big Data)* (pp. 51-57). IEEE.

[8] Yang, G., Jin, J., Lei, Q., Wang, Y., Zhou, J., Sun, Z., Li, X. and Wang, W., 2021, October. Garbage classification system with yolov5 based on image recognition. In *2021 IEEE 6th International Conference on Signal and Image Processing (ICSIP)* (pp. 11-18). IEEE.

[9] Kolla, N.S., Anumula, M., Sujana, S. and Ratnababu, M., 2023, May. Road Garbage Classification Using ResNet50. In *2023 International Conference on Signal Processing, Computation, Electronics, Power and Telecommunication (IConSCEPT)* (pp. 1-5). IEEE.

[10] Patil, A., Tatke, A., Vachhani, N., Patil, M. and Gulhane, P., 2021, August. Garbage classifying application using deep learning techniques. In *2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)* (pp. 122-130). IEEE.

[11] Ray, P.P., 2022. A review on TinyML: State-of-the-art and prospects. *Journal of King Saud University-Computer and Information Sciences*, 34(4), pp. 1595-1623.

[12] Kandel, I. and Castelli, M., 2020. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT express*, 6(4), pp.312-315.

[13] Lin, W., 2021, December. YOLO-green: A real-time classification and object detection model optimized for waste management. In *2021 IEEE International Conference on Big Data (Big Data)* (pp. 51-57). IEEE.

[14] Hermawan, I., Mardiyono, A., Iswara, R.W., Murad, F.A., Ardiawan, M.A. and Puspita, R., 2023, August. Development of Covid Medical Waste Object Classification System Using YOLOv5 on Raspberry Pi. In *2023 10th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)* (pp. 443-447). IEEE.