Jurnal Rekayasa Mekatronika (JRM)
Vol. 1 No. 2, September 2024, hal. 1 – 10 ISSN xxxx-xxxx

IMPLEMENTASI *GOOGLE COLABORATORY* PADA *TRAINING MODEL* UNTUK MENDETEKSI OBYEK PADA SISTEM SORTIR BARANG

[1]Muhammad Ghifari Apriansyah, [2]Adolf Asih Supriyanto, [3]Deni Kurnia
[1,2,3]Teknologi Rekayasa Mekatronika, Politeknik Enjinering Indorama, Purwakarta, Indonesia
[1]ghifariapap@gmail.com, [2]adolf@pei.ac.id, [3]deni.kurnia@pei.ac.id
co-author: [1]ghifariapap@gmail.com

Abstrak

Pengenalan obyek berbasis kecerdasan buatan dan machine learning mengalami perkembangan yang sangat pesat hingga saat ini. Salah satu bidang yang memanfaatkan teknologi ini adalah pengolahan citra. Pada penelitian ini menjelaskan tentang teknologi pengolahan citra yang diimplementasikan untuk mengenali barang pada sistem sortir berdasarkan data yang dilatih menggunakan Google Colaboratory dengan memanfaatkan TensorFlow Object Detection API dan model SSD Mobilenet V2. Aplikasi ini dirancang untuk mendeteksi dan mengukur akurasi dari input gambar yang telah dilatih, yaitu barang berdasarkan warna dan bentuk menggunakan metode menemukan obyek dalam gambar atau bisa disebut dengan object detection. Hasil dari pengujian menunjukkan bahwa sistem yang dikembangkan berhasil mencapai tingkat akurasi deteksi obyek dengan rata-rata sebesar 88,7% menggunakan analisa perhitungan confusion matrix, melalui pengujian pada empat jenis barang dengan tiga posisi yang berbeda.

Kata kunci: *Object Detection, Pengolahan Citra, Google Colaboratory, Sortir Barang, Confusion Matrix*

Abstract

Object recognition based on artificial intelligence and machine learning has undergone a very rapid development to this day. One of the areas that utilizes this technology is image processing. In this study, it explains about the image-processing technology implemented to recognize items on sorting systems based on data trained using Google Colaboratory using the TensorFlow Object Detection API and the Mobilenet V2 SSD model. The application is designed to detect and measure the accuracy of the image input that has been trained, that is items based on colors and shapes using the method of finding objects in images or can be called object detection. The results of the testing showed that the developed system achieved an average of 88,7% accuracy of object detection using matrix confusion calculation analysis, through testing on four types of objects with three different positions.

Keywords: *Object Detection, Image Processing, Google Colaboratory, Item Sorting, Confusion Matrix.*

Makalah dikirim 15 Februari 2024; Revisi 15 Maret 2024, Diterima 15 April 2024

1. PENDAHULUAN

Kecerdasan buatan (*Artificial Intelligence*) adalah istilah yang digunakan untuk menggambarkan kemampuan mesin dalam menampilkan tingkat kecerdasan yang menyerupai manusia [1]. Para peneliti saat ini telah banyak mengembangkan sistem yang memanfaatkan fitur untuk mengenali sebuah obyek atau hampir mendekati dengan kemampuan *visual* pada manusia [2]. Hal tersebut merupakan implementasi dari teknologi pengolahan citra atau suatu cabang ilmu yang bertujuan untuk meningkatkan kualitas gambar sehingga dapat diinterpretasikan oleh komputer [3]. Terkait dengan teknologi tersebut, pengolahan citra merupakan pondasi yang penting dalam teknologi *computer vision*, karena dapat mempelajari bagaimana cara komputer untuk mengambil informasi yang diperlukan dari gambar.

Perkembangan sistem kecerdasan buatan saat ini telah mempercepat kemajuan dalam penggunaan teknologi *computer vision* dan *image processing* [4]. Telah banyak peneliti yang melakukan penelitian mengenai implementasi dari teknologi tersebut yang diaplikasikan berdasarkan suatu kriteria tertentu. Diantaranya adalah penelitian mengenai identifikasi dan *tracking* obyek berbasis *image processing* secara *real time* [5] menggunakan *webcam* untuk melakukan pengambilan citra yang kemudian melakukan *tracking* obyek pada citra tersebut. Kemudian ada penelitian yang dilakukan [6] dengan memanfaatkan teknologi *computer vision* untuk mendeteksi dan menghitung jumlah manusia secara otomatis.

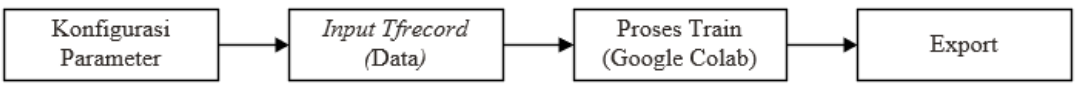
Selain itu, terdapat penelitian lain mengenai sistem kecerdasan buatan yang diimplementasikan menggunakan *framework TensorFlow object detection* API dan algoritma dari model *object detection* [7] untuk melakukan deteksi dan klasifikasi kendaraan bermotor. Kemudian satu tahun setelahnya terdapat pengembangan dari metode yang sama, yaitu penggunaan arsitektur SSD dengan menggunakan *pretrained model Mobilenet* yang digunakan untuk mengenali sebuah obyek bergerak menggunakan *input video capture* [8]. Penelitian terkini mengenai penggunaan arsitektur SSD *Mobilenet* yaitu pengenalan ekspresi wajah secara *real time* berbasis *Single Shot MultiBox Detector* [9]-[11] dalam metode *object detection*.

Berdasarkan referensi penelitian sebelumnya dan untuk mengisi *research gap* dari penelitian yang sudah dilakukan, maka akan dibuat sistem sortir barang dengan memanfaatkan teknologi pengolahan citra untuk mengenali barang yang akan disortir. *Framework* yang digunakan untuk melatih data dari barang yang akan dikenali dan dideteksi oleh sistem yaitu *TensorFlow object detection* API pada Google Colaboratory. Peran SSD dalam penelitian ini adalah mengatur proses deteksi obyek dengan pembuatan *bounding box*, sementara Mobilenet bertanggung jawab untuk mengekstrak fitur yang akan digunakan dalam proses klasifikasi [12]-[15].

2. METODE PENELITIAN

Dalam membangun penelitian ini, dilakukan beberapa tahapan. Setiap tahapan memiliki peran penting dalam memastikan bahwa perancangan dilakukan secara efektif dan efisien, serta dapat menghasilkan model deteksi obyek yang akurat dan efektif dalam mengenali bentuk dan warna dalam data gambar.

Model yang digunakan, terlebih dahulu melakukan proses pelatihan untuk diaplikasikan dengan metode *object detection* menggunakan *Google Colaboratory*, agar model mampu mendeteksi obyek dalam gambar atau vidio. Langkah pertama melibatkan persiapan *dataset* yang berisi gambar-gambar dengan obyek yang ingin dideteksi. Setelah itu, mengimpor *library* dan dependensi yang dibutuhkan, termasuk *TensorFlow Lite* dan *TensorFlow Object Detection* API, yang merupakan kerangka kerja yang kuat untuk deteksi obyek. Berikut merupakan skenario pengembangan model secara sederhana dapat dilihat seperti pada Gambar 1.



Gambar 1. *Proses labelling dan pelatihan data.*

Dalam proses pengolahan data untuk *Train Set*, pembuatan *tfrecord* dilakukan setelah selesai proses konfigurasi parameter. Berkas gambar yang sudah dilabeli dan berformat .jpg akan diubah menjadi *tfrecord* sebelum memulai pelatihan data. Konversi dari format .jpg ke *tfrecord* dilakukan pada Google Colaboratory dengan menggunakan program yang telah ditulis dalam *notebook*. *Dataset* tersebut perlu diproses secara memadai untuk keperluan pelatihan model, termasuk perubahan format gambar, pengkodean label, dan pembagian *dataset* menjadi bagian pelatihan dan validasi. Konfigurasi model SSD *MobileNet* V2 diatur sesuai dengan kebutuhan, seperti ukuran *batch*, langkah pembelajaran, dan parameter lainnya, yang kemudian disimpan dalam file konfigurasi. Terakhir, model yang telah dilatih dapat diekspor untuk digunakan dalam aplikasi pendeteksi barang berdasarkan bentuk dan warna.

2.1. Proses *Labelling*

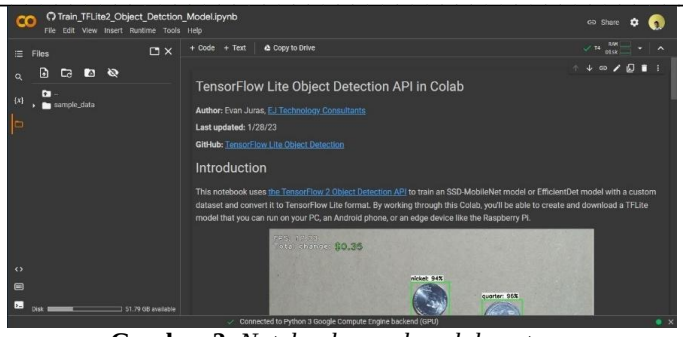
Pelabelan data merupakan langkah penting dalam pengembangan model *machine learning*, memastikan *dataset* yang dilatih memiliki label yang akurat dan relevan. Alat-alat pelabelan data, seperti *Labelling* yang memainkan peran penting dalam memfasilitasi proses ini, menggabungkan metode pelabelan manual dan otomatis. Proses pembuatan *labelling* tersebut dapat dilihat pada Gambar 2.

Gambar 2. *Proses labelling obyek.*

Proses pelabelan tersebut digunakan untuk memberi label pada gambar yang digunakan untuk model, dengan mengumpulkan beberapa gambar dalam *folder* untuk digunakan. Hasil dari proses tersebut akan mendapatkan *file* dengan ekstensi .XML yang berisikan koordinat dan ukuran dari model.

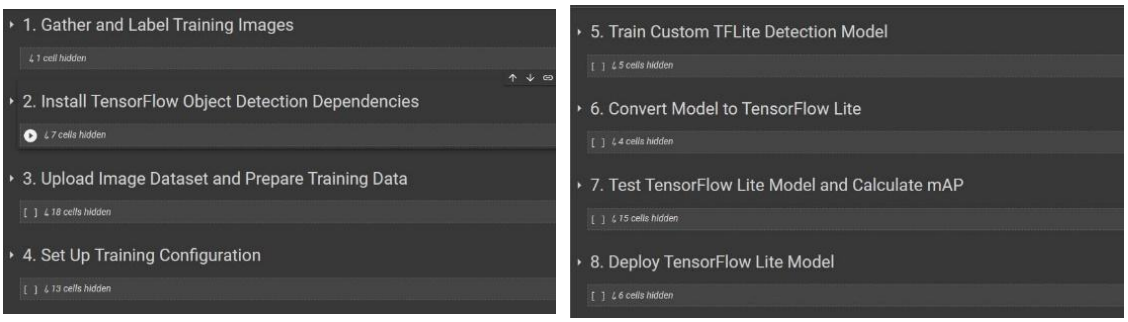
2.2. Proses *Training Model*

Training model adalah proses mengajari algoritma atau model pembelajaran mesin untuk memahami pola dan hubungan dalam data. Proses ini melibatkan pengoptimalan parameter model menggunakan *dataset*, yang merupakan kumpulan data yang terdiri dari *input* (fitur) dan *output* (label atau target). *Google Colaboratory* berperan dalam proses pelatihan model *object detection*. *Notebook* ini menggunakan API *Object Detection TensorFlow* 2 untuk melatih model SSD-*MobileNet* atau *model EfficientDet* dengan himpunan data kustom dan mengonversinya ke format *TensorFlow Lite*. Proses pelatihan model tersebut dapat dilihat pada Gambar 3.



Gambar 3. *Notebook google colaboratory.*

Setiap langkah dari proses pelatihan memiliki bagiannya sendiri. Dengan melakukan klik panah disamping judul untuk setiap bagian untuk memperluasnya, dan daftar isi dibagian kiri untuk melompat dari satu bagian ke bagian lainnya. Terdapat beberapa langkah untuk melakukan proses *training* pada model yang ingin digunakan seperti pada Gambar 4.



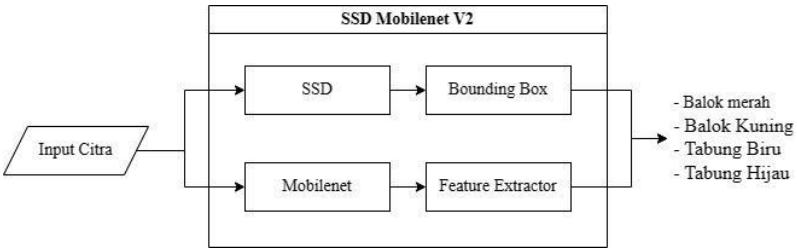
a b

Gambar 4. *Langkah proses train.*

Setelah selesai melakukan proses pelatihan maka akan didapatkan *file* dengan ekstensi .tflite dan labelmap.txt yang digunakan sebagai *input* model dari program untuk aplikasi *object detection* yang akan diintegrasikan dengan program sortir dan data yang ditampilkan berupa deteksi warna dan bentuk dari barang.

2.3. Aplikasi Deteksi Obyek

SSD Mobilenet berperan sebagai *base* model untuk deteksi obyek, menghasilkan *bounding box* untuk lokalisasi. *Mobilenet* digunakan sebagai *network* model untuk mengekstrak fitur guna mengklasifikasi obyek dalam gambar. Kombinasi SSD dan *Mobilenet* memperkuat aplikasi deteksi obyek dengan SSD menangani lokalisasi, sementara *Mobilenet* bertugas mengklasifikasi obyek seperti balok merah, balok kuning, tabung biru, dan tabung hijau. Proses aplikasi *object detection* tersebut dapat dilihat pada Gambar 5.

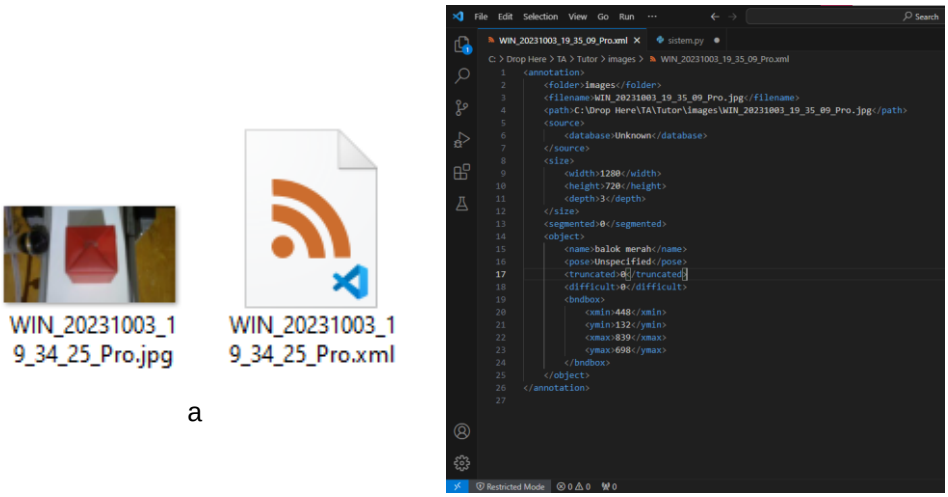


Gambar 5. *Skenario Penggunaan Model Object Detection.*

3. HASIL DAN PEMBAHASAN

3.1. Hasil *Labelling* dan *Training Model*

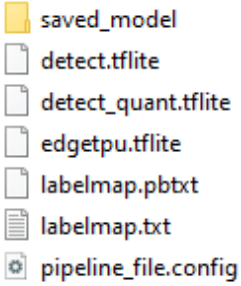
Pada hasil *labelling* data dan *training model* untuk membangun sistem secara keseluruhan. Terdapat *file* dengan ekstensi.xml yang berisi *script* untuk dijadikan model ketika proses *training* seperti pada Gambar 6 merupakan hasil ketika proses *labelling* sudah dilakukan, dan akan menghasilkan *file* .xml untuk *input* model ketika proses *training.*



Gambar 6. *File hasil labelling.*

Pada Gambar 7 merupakan *file* dari hasil *training Tensorflow Lite* yang sudah di*download* melalui *Google Colaboratory* untuk diintegrasikan dengan program dari sistem sortir. Terdapat beberapa *file* yang digunakan sebagai *input* model dalam aplikasi deteksi obyek, yaitu:

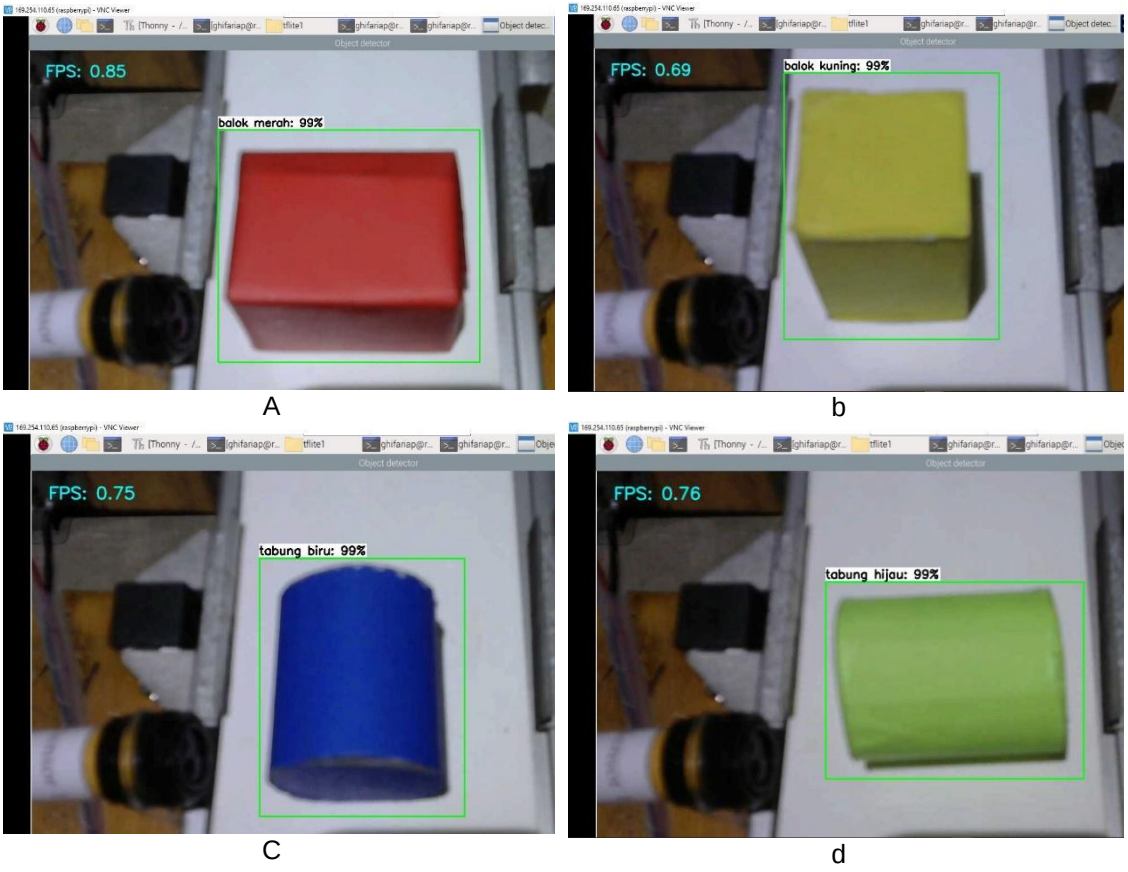
1. detect.tflite: *Model TensorFlow Lite* yang telah dioptimalkan untuk digunakan pada perangkat Raspberry.
2. detect_quant.tflite: Adalah versi kuantisasi dari model detect.tflite. Kuantisasi adalah proses mengurangi jumlah bit yang digunakan untuk merepresentasikan bobot dalam model, sehingga mengurangi ukuran model dan mempercepat inferensi pada perangkat dengan sumber daya terbatas.
3. edgetpu.tflite: Model tersebut dioptimalkan atau dikonversi untuk penggunaan *edge* TPU (*Tensor Processing Unit*), yang merupakan perangkat keras akselerator untuk tugas inferensi *machine learning*.
4. labelmap.txt: Berfungsi sebagai peta label yang mengaitkan kelas atau kategori yang ada dalam *dataset* dengan nomor atau indeks yang sesuai.



Gambar 7. *File dan folder hasil training.*

3.2. Hasil Aplikasi Deteksi Obyek

Pada Gambar 8 merupakan hasil inferensi dari model aplikasi deteksi obyek yang berhasil melakukan proses *training* pada *Google Colaboratory* dan diintegrasikan dengan program dari sistem sortir melalui Raspberry Pi 3B+. Terdapat empat *class* yang terdeteksi yaitu balok merah, balok kuning, tabung biru, dan tabung hijau. Keempat obyek tersebut mendapatkan tingkat *confidence* sebesar 99% yang dimana tingkat prediksi mendekati dengan *class*-nya.



Gambar 8. *Deteksi obyek.*

Gambar 8 di atas adalah hasil dari inferensi aplikasi deteksi obyek. Dari hasil tersebut, terdapat beberapa informasi penting yang yang akan dijelaskan sebagai berikut.

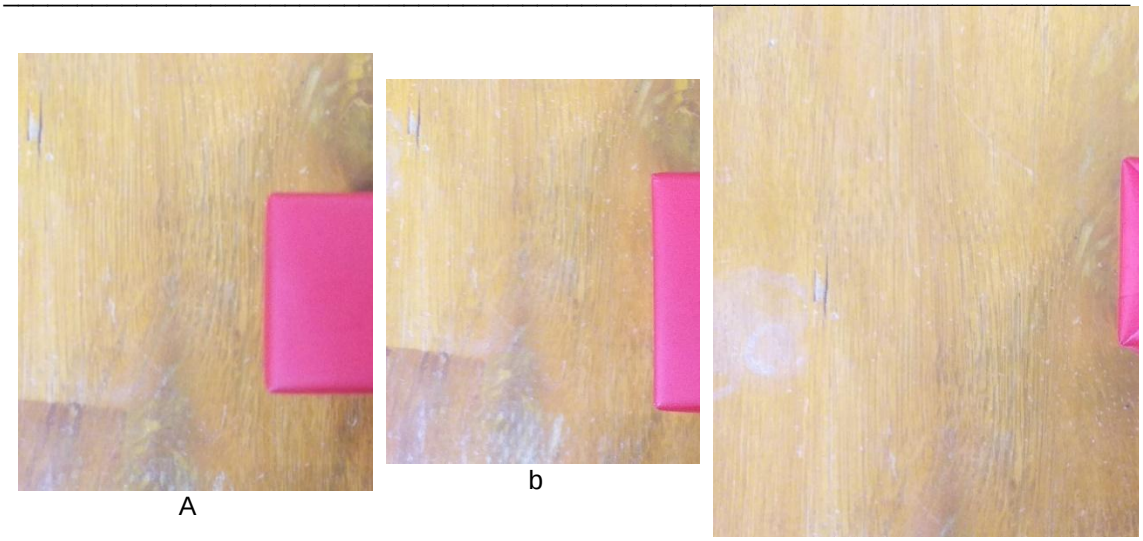
1. *Capture image* oleh kamera: adalah *image* atau citra yang saat itu sedang diambil oleh kamera.
2. Data FPS: mengacu pada *Frames Per Second*, yaitu seberapa cepat perangkat mampu menghasilkan bingkai *image* dalam satu detik.
3. Hasil deteksi: adalah informasi terkait dengan hasil deteksi obyek yang sedang di-*capture*. Hasil deteksi tersebut dapat mencakup kategori atau label yang menjelaskan apa yang terdapat pada *image* tersebut.
4. Nilai *confidence*: adalah indikasi tingkat keyakinan sistem terhadap hasil deteksi yang diberikan. Ketika nilai *confidence* mencapai 99%, maka sistem sangat yakin bahwa hasil dari deteksi yang dilakukan benar dan sesuai.

3.3. Pengujian Aplikasi Deteksi Obyek

Pengujian pada aplikasi deteksi obyek mencakup beberapa parameter yaitu seberapa cepat model dapat menghasilkan *output detection* dengan menghitung nilai pada *confidence* dengan persentase yang diukur. Selanjutnya untuk mengukur tingkat kecepatan dalam satuan FPS, yang menunjukkan seberapa banyak *frame* gambar yang dapat diproses dalam satu waktu. Dan mengukuran kecepatan proses inferensi waktu yang diperlukan oleh model untuk melakukan inferensi pada data input tertentu dalam parameter *inference time*. Pada Tabel 1 merupakan data dari nilai parameter FPS, *confidence*, dan *inference time* untuk setiap kategori dan pada Gambar 9 merupakan posisi dari tiap barang yang akan dideteksi oleh kamera.

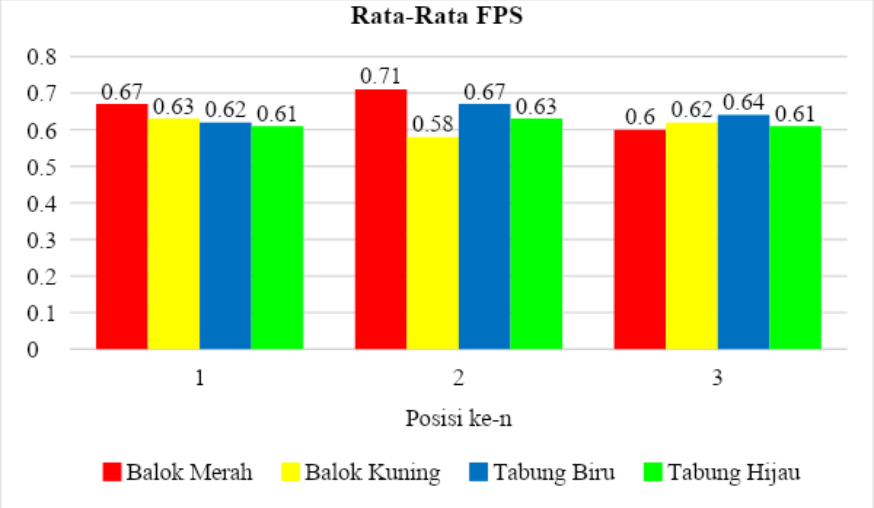
Tabel 1. *Pengujian Aplikasi Deteksi Obyek*

Kategori Barang		Rata Rata FPS	Rata Rata *Confidence* (%)	Rata Rata *Inference* (ms)
Balok Merah	Posisi 1	0,67	99%	1478,27
	Posisi 2	0,71	99%	1368,08
	Posisi 3	0,6	99%	1688,44
Balok Kuning	Posisi 1	0,63	99%	1562,65
	Posisi 2	0,58	99%	1672,98
	Posisi 3	0,62	99%	1565,81
Tabung Biru	Posisi 1	0,62	99%	1587,83
	Posisi 2	0,67	99%	1483,08
	Posisi 3	0,64	99%	1518,71
Tabung Hijau	Posisi 1	0,61	99%	1587,44
	Posisi 2	0,63	99%	1563,3
	Posisi 3	0,61	99%	1605,93



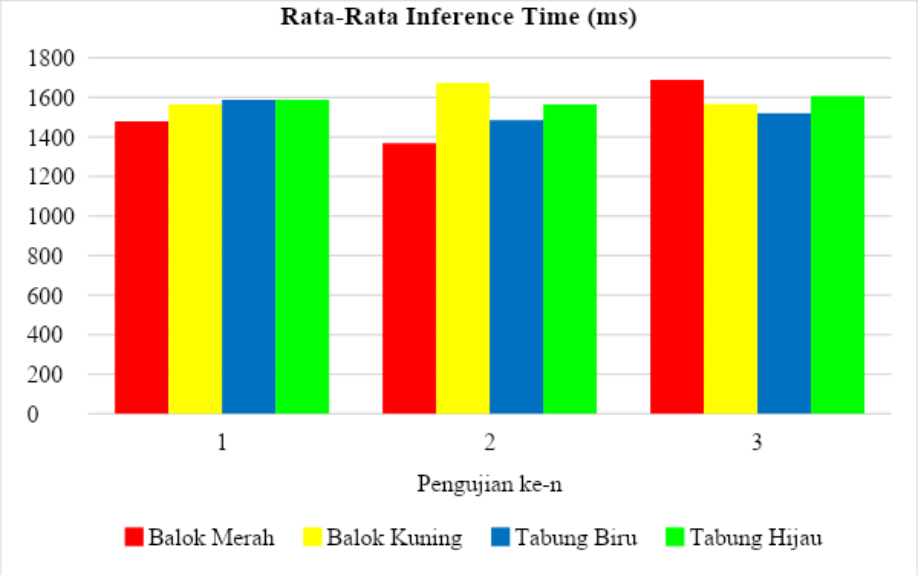
Gambar 9. *Barang Posisi 1, 2, dan 3.*

Gambar 10 merupakan hasil pengujian dari aplikasi deteksi obyek yang dimuat dalam bentuk grafik. Dilakukan pengujian sebanyak 100 kali percobaan tiap masing-masing kategori dan posisi. Dapat diambil kesimpulan bahwa dalam menguji performa *Raspberry Pi* untuk implementasi metode *Object Detection* ini dapat menghasilkan pendeteksian dengan nilai FPS yang berada diangka 0,5 sampai 1 FPS dan didapatkan hasil dengan rata-rata sebesar 0,64 FPS. Disetiap kategori terdapat pendeteksian dengan 3 posisi berbeda yang dimana setiap posisinya menunjukan kenaikan pada hasil FPS dengan nilai yang berubah-ubah disetiap kategorinya. Berdasarkan hasil tersebut, bahwa performa aplikasi dapat dipengaruhi oleh beberapa faktor utama, yaitu *Raspberry Pi* yang digunakan memiliki sumber daya terbatas, termasuk CPU, RAM, dan GPU, sehingga model yang digunakan tidak cukup kuat untuk menangani tugas deteksi obyek yang memerlukan komputasi intensif. Selain itu, penggunaan model deteksi obyek yang kompleks dan berat dapat memberikan beban berlebih pada *Raspberry Pi*.



Gambar 10. *Grafik rata-rata FPS.*

Gambar 11 merupakan hasil pengujian Untuk *inference time* disetiap kategori dan posisi. Didapatkan hasil dengan rata-rata rentang waktu inferensi antara 1 hingga 1,8 detik. Rentang waktu tersebut menunjukkan bahwa sistem mampu memberikan hasil deteksi dalam batas waktu yang dapat diterima, sesuai dengan persyaratan aplikasi deteksi obyek secara *real-time*. Pada setiap kategori terdapat beberapa perbedaan hasil yang didapatkan, untuk kategori balok merah, tabung biru, dan tabung hijau terdapat kenaikan waktu inferensi yang dimana hal tersebut menunjukan beberapa masalah dalam kinerja aplikasi deteksi obyek. Faktor yang mempengaruhi kenaikan inferensi waktu tersebut yaitu penggunaan model deteksi obyek yang berat dan kompleks, hingga berakibat pada hasil dari pendeteksian.



Gambar 11. *Rata-rata inference time.*

4. KESIMPULAN

Berdasarkan hasil dan pembahasan pada penelitian ini maka dapat diperoleh kesimpulan sebagai berikut:

1. Penggunaan *Google Colaboratory* sebagai *platform* untuk melatih model kecerdasan buatan dalam mendeteksi obyek pada sistem sortir barang.
2. Penggunaan teknologi *image processing* dalam sistem sortir barang dengan menggunakan teknik-teknik seperti deteksi warna dan bentuk. Sistem dapat secara efisien mengklasifikasikan barang-barang berdasarkan karakteristik visualnya. Hal tersebut memungkinkan otomatisasi proses sortir barang dengan tingkat akurasi yang tinggi.
3. Sistem sortir berdasarkan bentuk dan warna berjalan sesuai dengan perencanaan. Dan hasil dari pengujian sistem dapat mengenali obyek pada gambar sesuai dengan kategori yang diatur pada *dataset* ketika pelatihan.

.

DAFTAR PUSTAKA

[1] A. Roihan, P. A. Sunarya, and A. S. Rafika, “Pemanfaatan Machine Learning dalam Berbagai Bidang: Review paper,” *IJCIT (Indonesian J. Comput. Inf. Technol.*, vol. 5, no. 1, pp. 75–82, 2020, doi: 10.31294/ijcit.v5i1.7951.

[2] T. Susim and C. Darujati, “Pengolahan Citra untuk Pengenalan Wajah (Face Recognition) Menggunakan OpenCV,” *J. Syntax Admiration*, vol. 2, no. 3, pp. 534–545, 2021, doi: 10.46799/jsa.v2i3.202.

[3] A. Chairi and R. Mukhaiyar, “Sistem Kontrol Color Sorting Machine dengan Pengolahan

Citra Digital," *JTEIN J. Tek. Elektro Indones.*, vol. 4, no. 1, pp. 387–396, 2023.

[4] D. S. Pratama, L. Anifah, L. Rakhmawati, and R. H. P. A. T., "Rancang Bangun Conveyor Penyortir Mur Berbasis Raspberry Pi Menggunakan Metode Contour Area," *J. Tek. Elektro*, vol. 11, no. 2, pp. 246–247, 2022.

[5] H. Mulyawan, M. Z. H. Samsono, and Setiawardhana, "Identifikasi Dan Tracking Obyek Berbasis Image," *Identifikas Dan Track. Obyek Berbas. Image Process. Secara Real Time*, pp. 1–5, 2011.

[6] T. A. Dompeipen and S. R. U. Sompie, "Penerapan computer vision untuk pendeteksian dan penghitung jumlah manusia," *J. Tek. Inform.*, vol. 15, no. 4, pp. 1–12, 2020.

[7] I. Maryati, "Website Perpustakaan 'Library HUB' dengan Pencarian Buku Berdasarkan Gambar Menggunakan Google MLKit," *JATISI (Jurnal Tek. Inform. dan Sist. Informasi)*, vol. 8, no. 4, pp. 1821–1831, 2021, doi: 10.35957/jatisi.v8i4.1269.

[8] K. Falah, M. Gustiana H, and U. Ungkawa, "Karakteristik Metode Mobilenet-SSD Dengan Pre-Trained Model Mobilenet Untuk Obyek Bergerak," *Pros. Disem. FTI Ganjil*, vol. X, no. X, pp. 1–13, 2022.

[9] M. Rifqi, D. Ulhaq, D. Firdaus, and M. A. Zaidan, "Pengenalan Ekspresi Wajah Secara Real-Time Menggunakan Metode SSD Mobilenet Berbasis Android," vol. 5, no. 1, 2023.

[10] Azizah Azzahra, Fitri Elvira Ananda, "Rancang Bangun Sistem Kehadiran Secara Real Time Menggunakan Face Recognition Dengan Metode SSD di SMK Negeri 53 Jakarta," JITET (Jurnal Informatika dan Teknik Elektro Terapan), Vol. 12 No. 1, 664 – 675, 2024.

[11] Ridwan Gunadi Fajri, Imam Santoso dan Yosua Alvin Adi Soetrisno,"Perancangan Program Pendeteksi dan Pengklasifikasi Jenis Kendaraan dengan Metode Convolutional Neural Network (CNN) Deep Learning," TRANSIENT, Vol. 9, No. 1, 97 – 106, 2020.

[12] Ida Astuti, Winda Widya Ariestya, Bambang Solehudin, "Deteksi Obyek Daun Semanggi Secara Real Time Menggunakan CNN-Single Shot Multibox Detector (SSD)," JURNAL ILMIAH FIFO, Volume XIV, No.1, hal. 47 – 58, 2022.

[13] Muhammad Rifqi Daffa Ulhaq, dkk." Pengenalan Ekspresi Wajah Secara Real-Time Menggunakan Metode SSD Mobilenet Berbasis Android,"Journal of Technology Informatics (JoTI), Vol.5, No.1, Halm. 48-52, 2023.

[14] Ivan Besando Pakpahan, Ika Candra Dew, "Pendeteksian Lubang Pada Jalanan Menggunakan Metode SSD-MobileNet," Indonesian Journal of Electronics and Instrumentation Systems (IJEIS) Vol.11, No.2, pp. 213-222, , 2021.

[15] P. R. Aningtyas, A. Sumin, and S. Wirawan, "Pembuatan Aplikasi Deteksi Obyek Menggunakan TensorFlow Object Detection API dengan Memanfaatkan SSD MobileNet V2 Sebagai Model Pra - Terlatih," *J. Ilm. Komputasi*, vol. 19, no. 3, pp. 421–430, 2020, doi: 10.32409/jikstik.19.3.68.