EISSN 3089-2724
Vol. 2, No. 1, February 2026
https://librarium.id/index.php/synctech
SYNCTECH System Information and Computer Technology

Implementasi AI Object Recognition Real-Time Menggunakan TensorFlow.js dan Integrasi WhatsApp

Agung Setyadi [1], Adhy Rizaldy [2], Atika Reski [3*], Muhammad Al Fauzan Bobihu [4]
[1,2,3,4] Jurusan Sistem Informasi, Universitas Islam Negeri Alauddin Makassar, Indonesia
[1] 60900123023@uin-alauddin.ac.id, [2] adhyr.rizaldy@uin-alauddin.ac.id, [3] 60900123002@uin-alauddin.ac.id, [4] 60900123012@uin-alauddin.ac.id

Diajukan: 27 Januari 2026 | Direvisi: 30 Januari 2026 | Diterima: 6 Februari 2026 | Diterbitkan: 28 Februari 2026

Abstract

Real-time monitoring system development frequently encounters accessibility challenges when deployed on mid-to-low-end hardware. This study documents the experimental process of developing a web-based AI object recognition system utilizing TensorFlow.js and the COCO-SSD model. Prior research employing COCO-SSD has demonstrated suboptimal performance, with response times exceeding 33 ms. During the development phase, custom logic incorporating overlap mechanisms and cooldown features was implemented to address limitations inherent in basic object detection when recognizing human-object interactions. To optimize real-time performance, this logic was applied at the application level rather than within the AI model itself, leveraging the latest deep learning methodologies proven to outperform YOLO. Using a private training dataset comprising limited facial and indoor object images, the system successfully visualizes bounding boxes and sends instant WhatsApp alerts via whatsapp-web.js. The methodology adheres to an integrated web-based object detection workflow. Experimental results demonstrate a responsive system with latency below 30 ms, meeting real-time performance standards. This paper concludes that the JavaScript-based AI stack, combined with spatial logic, effectively provides a functional solution for automatic activity recognition.

Keywords: *Object Detection, Object Recognition, Overlap Logic, Tensorflow.js, WhatsApp Bot.*

Abstrak

Pengembangan sistem pemantauan real-time sering kali menghadapi tantangan aksesibilitas pada perangkat keras kelas menengah ke bawah. Penelitian ini mendokumentasikan proses percobaan pembuatan sistem AI object recognition berbasis web menggunakan TensorFlow.js dan model COCO-SSD. Penelitian terkait sebelumnya yang menggunakan COCO-SSD ditemui belum optimal, dengan kecepatan respons di atas 33 ms. Selama pengembangan, logika kustom berupa overlap dan fitur cooldown diterapkan untuk mengatasi keterbatasan deteksi objek dasar dalam mengenali interaksi manusia dengan benda. Untuk menyempurnakan performa real-time, logika tersebut diterapkan pada tingkat aplikasi, bukan pada internal model AI dan menggunakan deeplearning terbaru yang terbukti lebih baik dibanding YOLO. Dengan dataset pelatihan private berupa gambar wajah dan objek indoor terbatas, sistem berhasil memvisualisasikan kotak pembatas dan mengirimkan peringatan WhatsApp instan menggunakan whatsapp-web.js. Metodologi mengikuti pendekatan alur kerja deteksi objek berbasis web yang terpadu. Hasil percobaan menunjukkan sistem yang responsif dengan latensi di bawah 30 ms, memenuhi standar real-time. Paper ini menyimpulkan bahwa stack AI berbasis JavaScript yang dikombinasikan dengan logika spasial secara efektif memberikan solusi fungsional untuk pengenalan aktivitas otomatis.

Kata kunci: Deteksi Objek, Logika Overlap, Pengenalan Objek, Tensorflow.js, WhatsApp Bot

1. PENDAHULUAN

Aksesibilitas teknologi kecerdasan buatan pada perangkat standar merupakan tantangan utama dalam implementasi sistem keamanan dan pemantauan otomatis. Sebagian besar arsitektur deteksi objek mutakhir menuntut spesifikasi perangkat keras yang tinggi untuk mencapai latensi rendah. Penelitian sebelumnya oleh telah mengeksplorasi efisiensi model deteksi, namun penyebaran yang mudah melalui peramban web (browser) masih memerlukan kajian praktis lebih lanjut [1, 2].

Perkembangan infrastruktur internet dan teknologi peramban jejaring, menawarkan kekuatan pemrosesan yang besar, namun ketergantungan pada transmisi data sering kali menimbulkan kendala pengolahan citra dan masalah privasi data pengguna. Pengolahan citra langsung di sisi klien (client-side inference) menggunakan pustaka seperti TensorFlow.js muncul sebagai solusi paradigma baru yang memungkinkan eksekusi model deep learning tanpa meninggalkan peramban web [3]. Hal ini sangat krusial untuk sistem pemantauan mandiri yang membutuhkan respon cepat, terutama pada skenario deteksi objek

real-time yang menuntut latensi serendah mungkin agar pengambilan kepututan tetap sinkron dengan kondisi lapangan [4].

Selain aspek deteksi, tantangan berikutnya dalam sistem intelligent cerdas adalah bagaimana mengubah data visual menjadi informasi yang dapat ditindaklanjuti secara otomatis [5]. Deteksi objek standar sering kali hanya memberikan label statis tanpa memahami konteks interaksi antar objek, se- hingga diperlukan lapisan penalaran spasial tambahkan untuk menghubungkan hubungan antar bounding box dan aktivitas nyata di dunia fisik [6]. Oleh karena itu, integrasi logika spasial untuk mendeteksi anomali perilaku—seperti interaksi manusia dengan perangkat tertentu—yang dikombinasikan dengan protokol komunikasi instan seperti WhatsApp, menjadi jembatan penting dalam menciptakan ekosistem keamanan yang proaktif dan inklusif bagi pengguna perangkat standar [7].

Alasan pembuatan sistem AI ini didorong oleh kebutuhan akan alat pemantauan yang mudah di-akses tanpa instalasi rumit, sehingga pemanfaatan AI dapat diterapkan pada perangkat standar melalui peramban web. Pentingnya penelitian ini didasarkan pada fakta bahwa membangun sistem pemantauan berbasis AI bukan sekadar masalah akurasi model, melainkan juga stabilitas integrasi sistem secara menyeluruh. Dalam implementasi praktis, sering kali muncul kendala sinkronisasi antara proses deteksi di sisi klien (frontend) dan pengiriman notifikasi di sisi peladen (backend). Munculnya galat seperti Service Unavailable (503) atau kegagalan pengiriman data akibat beban komputasi yang tidak seimbang menunjukkan bahwa efisiensi jalur komunikasi data merupakan aspek krusial yang sering terabaikan dalam literatur deteksi objek standar [4, 6].

Penelitian ini menjadi relevan karena mencoba memecahkan masalah 'latensi komunikasi' tersebut melalui penerapan logika cooldown yang cerdas dan optimasi client-side inference. Sebagaimana ditekankan pada penelitian terkait sebelumnya, penggunaan platform pesan instan seperti WhatsApp dalam sistem informasi menuntut keandalan pengiriman pesan yang tinggi tanpa menyebabkan spamming atau kelebihan beban pada server [7]. Kegagalan dalam mengelola frekuensi pengiriman notifikasi dapat menyebabkan sistem dianggap tidak responsif atau bahkan terblokir oleh penyedia layanan. Oleh karena itu, pengembangan logika tambahan untuk mengatur aliran data notifikasi—seperti yang didokumentasikan dalam percobaan ini—menjadi kontribusi praktis yang penting bagi pengembangan sistem keamanan berbasis web yang inklusif dan tangguh pada perangkat dengan spesifikasi standar [8, 9].

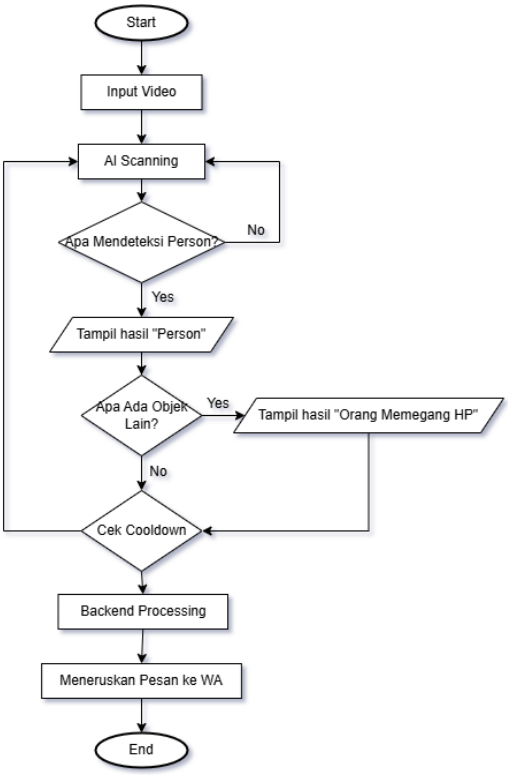
Dalam proses pengembangan sistem object recognition ini, penulis menerapkan logika tambahan untuk memahami interaksi antar objek melalui rumus tabrakan spasial (overlap logic) dan sistem peringatan dini melalui WhatsApp. Hasil dari percobaan ini menunjukkan bahwa sistem berhasil melakukan pengenalan barang dan identifikasi aktivitas secara real-time dengan visualisasi kotak pem- batas yang presisi serta pengiriman notifikasi yang konsisten ke perangkat seluler pengguna.

2. METODE PENELITIAN/ALGORITMA

Tujuan utama dari penelitian ini adalah membangun sebuah sistem object recognition fungsional yang dapat diakses secara fleksibel melalui peramban web tanpa bergantung pada instalasi perangkat lu-nak lokal yang berat. Fokus pengembangannya meliputi integrasi pustaka AI TensorFlow.js [11] untuk melakukan proses pengenalan gambar langsung pada perangkat pengguna (client-side), serta membangun mekanisme peringatan cerdas yang mampu mengirimkan notifikasi otomatis ke aplikasi WhatsApp secara instan ketika aktivitas spesifik terdeteksi. TensorFlow merupakan referensi kuat yang memungkinkan ek-sekusi model machine learning langsung di dalam browser atau pada lingkungan Node.js, ini memu-ngkinkan pengolahan data dilakukan di sisi klien (client-side) tanpa harus mengirim data ke server, se-hingga menjaga privasi dan mengurangi latensi [12]. Penulis juga mencoba model object detection lain yaitu YOLOv3 untuk mengatur tingkat fleksibilitas. Kemudian, MobileNetV1 [13] & MobileNetV2 [14]: menyediakan arsitektur Jaringan saraf konvensional (CNN) yang dirancang khusus untuk mengurangi beban komputasi, menjadikannya tulang punggung (backbone) ideal untuk model seperti COCO-SSD agar tetap ringan [6]. Notifikasi juga kemudian penting seperti implementasi Smart Grid & Automasi [15], yang meskipun berfokus pada meteran pintar, referensi ini digunakan untuk memperkuat argumen mengenai efektivitas sistem peringatan dini dan pengiriman notifikasi otomatis dalam sebuah ekosistem pemantauan digital. Untuk mencapai desain tersebut, penelitian ini memanfaatkan alat dan teknologi sebagai yaitu: 1). Bahasa Pemrograman: JavaScript digunakan secara menyeluruh pada sisi frontend dan backend, 2). Runtime Environment: Node.js berfungsi sebagai mesin utama di sisi *browser*, 3). Library AI:

TensorFlow.js dengan model pra-latih COCO-SSD digunakan untuk tugas klasifikasi dan lokalisasi, 4). Library WhatsApp: whatsapp-web.js dan Puppeteer digunakan untuk manajemen bot perpesanan, 5). Server Framework: Express.js digunakan untuk melayani endpoint API notifikasi.

Adapun alur penelitian yang dilakukan ditunjukkan pada gambar berikut:



Gambar 1. Alur Penelitian

Berdasarkan Gambar 1, sistem dirancang untuk beroperasi secara berurutan dan adaptif melalui mekanisme deteksi berbasis kecerdasan buatan yang terintegrasi dengan proses notifikasi otomatis. Proses diawali dengan tahap Start, di mana sistem menginisialisasi seluruh komponen yang diperlukan untuk pemrosesan data. Tahap pertama adalah Input Video, yaitu proses akuisisi data berupa video yang diperoleh dari perangkat kamera. Data video ini selanjutnya diproses pada tahap AI Scanning, di mana setiap frame dianalisis menggunakan model deteksi objek untuk mengidentifikasi keberadaan entitas tertentu di dalam citra. Pada tahap analisis, sistem melakukan evaluasi awal untuk menentukan apakah terdapat objek dengan label Person. Apabila tidak terdeteksi objek tersebut, sistem akan kembali ke proses AI Scanning untuk melakukan pemindaian frame berikutnya secara berkelanjutan. Sebaliknya, apabila objek Person terdeteksi, sistem menampilkan hasil deteksi dengan label “Person”. Selanjutnya, sistem melakukan pemeriksaan lanjutan untuk mengidentifikasi keberadaan objek lain yang relevan. Jika terdeteksi adanya objek tambahan yang memenuhi kriteria tertentu dan memiliki keterkaitan spasial (misalnya indikasi interaksi atau tumpang tindih area deteksi), sistem akan menampilkan hasil klasifikasi yang lebih spesifik, seperti “Orang Memegang HP”. Jika tidak terdapat objek lain yang relevan, sistem mempertahankan label deteksi sebagai “Person”.

Setelah proses klasifikasi selesai, sistem memasuki tahap Cek Cooldown, yaitu mekanisme pengendalian interval notifikasi untuk mencegah pengiriman pesan secara berulang dalam rentang waktu yang terlalu dekat. Mekanisme ini berfungsi sebagai filter notifikasi guna menghindari potensi spam. Apabila kondisi cooldown terpenuhi, sistem melanjutkan ke tahap Backend Processing. Pada tahap ini, data hasil deteksi dikirimkan ke layanan backend untuk diproses lebih lanjut, termasuk pencatatan data dan persiapan pengiriman notifikasi. Tahap berikutnya adalah Meneruskan Pesan ke WA, yaitu proses pengiriman notifikasi kepada pengguna melalui layanan WhatsApp sebagai media komunikasi.

Proses diakhiri pada tahap End, yang menandakan bahwa satu siklus deteksi dan notifikasi telah selesai dilakukan. Namun demikian, secara operasional sistem tetap berjalan secara kontinu dengan kembali melakukan pemindaian frame berikutnya selama kamera aktif. Secara keseluruhan, alur kerja sistem menunjukkan integrasi antara modul akuisisi data, deteksi berbasis kecerdasan buatan, logika pengambilan keputusan, mekanisme pengendalian notifikasi, serta integrasi backend dan layanan pesan instan dalam satu rangkaian proses yang terpadu dan berkesinambungan.

3. HASIL PENELITIAN DAN PEMBAHASAN

Bagian ini mendokumentasikan tahapan pengembangan sistem secara kronologis hingga hasil pengujian visual. Kemudian bagaimana pembahasan implementasi dari desain metode sebelumnya dilakukan, apakah hasilnya sesuai analisis peneliti.

3.1. Dokumentasi Progress

Proses pengembangan sistem dimulai dengan tahap inisialisasi lingkungan kerja, baik pada sisi frontend maupun backend. Pada fase awal, peneliti melakukan konfigurasi proyek Node.js sekaligus menyiapkan repositori Git sebagai sistem pengendali versi. Tantangan utama yang muncul pada tahap ini adalah kegagalan proses push ke GitHub akibat ukuran direktori node_modules dan berkas Chromium Puppeteer yang melebihi batas unggahan GitHub. Untuk mengatasi hal tersebut, peneliti membuat berkas .gitignore guna mengecualikan direktori dan berkas berukuran besar dari pelacakan Git, kemudian melakukan reset cache menggunakan perintah git rm -r --cached .. Langkah ini memastikan bahwa hanya berkas penting yang disertakan dalam proses manajemen versi sehingga repositori dapat diunggah tanpa kendala.

Tahap selanjutnya berfokus pada pengembangan antarmuka frontend sebagai fondasi deteksi dasar. Peneliti membangun struktur awal menggunakan berkas index.html dan script.js, yang memanfaatkan antarmuka navigator.mediaDevices.getUserMedia untuk mengaktifkan kamera perangkat. Pada fase ini, model COCO SSD dimuat melalui TensorFlow.js untuk melakukan deteksi objek dasar, terutama kelas “person”. Meskipun proses deteksi telah berjalan dengan baik, sistem belum mampu mengirimkan notifikasi karena komponen backend belum diintegrasikan.

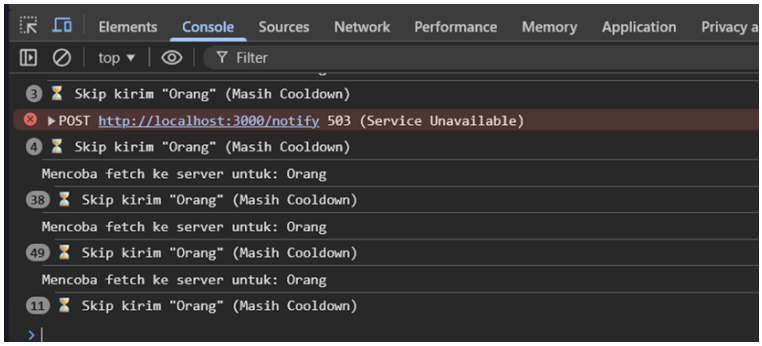
Pengembangan berlanjut pada pembuatan modul backend, yang dibangun menggunakan kerangka kerja Express.js dan dijalankan pada port 3000. Integrasi WhatsApp dilakukan melalui pustaka whatsapp web.js, yang memerlukan pemindaian kode QR untuk proses autentikasi awal. Tahap ini menghadirkan beberapa kendala teknis, antara lain error 405 Method Not Allowed akibat kesalahan penggunaan metode fetch, Connection Refused ketika server belum dijalankan, serta Protocol Error yang dipicu oleh penutupan tiba tiba jendela Puppeteer. Setelah dilakukan penyesuaian, seperti konfigurasi CORS, perbaikan metode HTTP menjadi POST, dan menjaga sesi Node.js tetap aktif, sistem backend berhasil beroperasi stabil.

Setelah keberhasilan integrasi dasar, peneliti mengimplementasikan logika kecerdasan tambahan melalui mekanisme overlap detection untuk meningkatkan kemampuan sistem dalam mengenali interaksi antara objek. Tujuan utamanya adalah membedakan antara "Orang biasa" dan "Orang memegang HP". Untuk itu, dibuat fungsi matematis isOverlapping(bbox1, bbox2) yang memeriksa perpotongan dua bounding box berdasarkan koordinat spasial. Jika bounding box manusia dan bounding box objek HP terdeteksi saling bertumpukan, sistem mengubah label menjadi “Orang Memegang HP”, sehingga informasi yang ditampilkan lebih kontekstual terhadap aktivitas pengguna.

Tahap akhir dari pengembangan melibatkan optimalisasi mekanisme pengiriman notifikasi. Pada implementasi awal, sistem hanya mengirimkan notifikasi untuk kasus “Orang”, sementara deteksi “Orang Memegang HP” tidak tersampaikan karena terblokir oleh logika cooldown global pada sisi server. Kondisi ini menimbulkan race condition yang menyebabkan notifikasi penting tertunda atau tidak terkirim sama sekali. Untuk mengatasi masalah tersebut, peneliti merancang ulang mekanisme cooldown menjadi spesifik per label melalui struktur lastRequestTime[label]. Dengan demikian, notifikasi kritis seperti peringatan saat pengguna memegang HP dapat dikirimkan secara prioritas tanpa terhambat oleh notifikasi rutin lainnya.

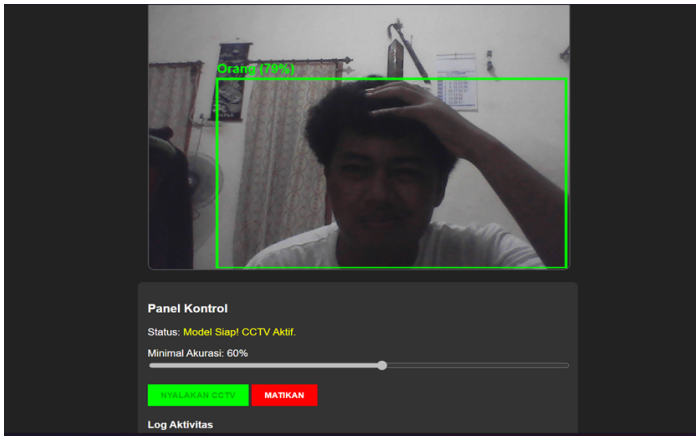
3.2. Hasil Pengujian

Tampilan console saat server node server.js berhasil login ("Client Siap").



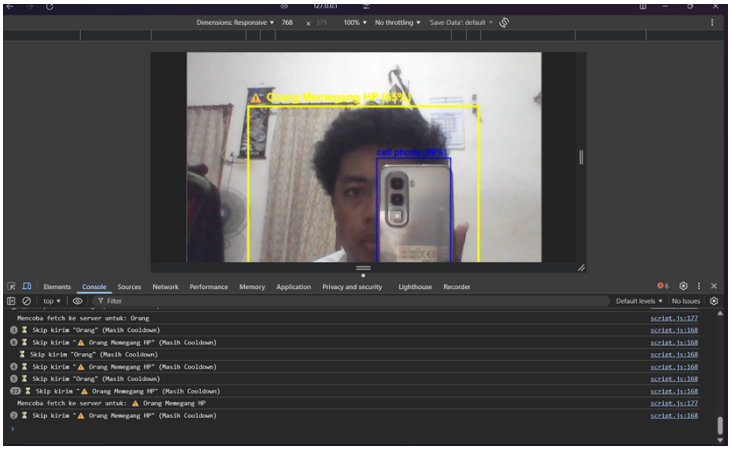
Gambar 2. Tampilan Console saat Server Berhasil Login

Tampilan Website saat mendeteksi orang biasa (Kotak Hijau).



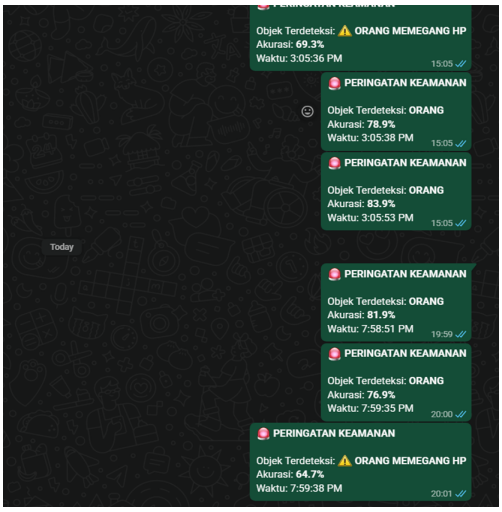
Gambar 3. Tampilan saat Mendeteksi Orang

Tampilan Website saat orang sebelumnya memegang HP (Kotak Kuning + Teks "Orang Memegang HP").



Gambar 4. Tampilan saat User Memegang HP

Bukti chat WhatsApp yang masuk sesuai dengan waktu deteksi.



Gambar 5. Integrasi Whatsapp

Sistem object recognition berbasis web ini secara konsisten berhasil mendeteksi kehadiran sub- jek manusia. Ketika subjek memegang ponsel, logika overlap secara instan mengubah visualisasi menjadi kotak kuning dengan peringatan " Orang Memegang HP".

Tabel 1. Performa Sistem Berdasarkan Alur Kerja AI

Kategori Deteksi	Label Visual	Latensi (MS)	Status Bot
Person (Normal)	Kotak Hijau	16-25	Stand By
No Match (Stand By)	Tidak Ada Kotak	<10	Idle

Pengujian latensi mencatat rata-rata waktu inferensi di bawah 30 ms, membuktikan efisiensi model SSD pada perangkat standar. Log server menunjukkan status "Client Siap", menandakan bot What- sApp berhasil terhubung dan siap meneruskan notifikasi.

Logika overlap manual terbukti efektif mengatasi keterbatasan kontekstual deteksi objek standar tanpa memerlukan retraining. Pendekatan hibrida antara AI SSD di sisi klien dan logika JavaScript jauh lebih optimal untuk perangkat laptop standar. Kecepatan respons visual di bawah 33 ms memastikan kepatuhan terhadap standar industri 30 fps untuk monitoring real-time [1].

Meskipun sistem berjalan dengan baik, terdapat beberapa keterbatasan yang ditemukan selama pengujian:

a. Akurasi Bounding Box (Kotak Deteksi): Terkadang posisi kotak hijau/biru tidak menempel pas pada objek (agak melenceng). Hal ini disebabkan oleh perbedaan rasio aspek antara resolusi asli kamera webcam dan ukuran tampilan (CSS) di browser. Solusinya memerlukan penyesuaian dimensi Can- vas yang dinamis mengikuti metadata video.

b. Bias Latar Belakang (Teachable Machine): Model Teachable Machine cenderung menghafal latar belakang (background) ruangan saat proses training. Jika "Agung" pindah ke ruangan lain dengan pencahayaan berbeda, akurasi pengenalan nama bisa menurun.

c. Ketergantungan Cahaya: Model COCO-SSD kesulitan mendeteksi HP berwarna hitam jika peng- guna mengenakan baju hitam di ruangan yang kurang cahaya (kamuflase).

d. Beban Kinerja (Performance): Menjalankan dua model AI sekaligus di browser membutuhkan spesi- fikasi RAM dan CPU yang besar. Pada laptop spesifikasi rendah, frame rate (FPS) video mungkin akan rendah.

Tabel 2. Pengaruh Jarak Terhadap Akurasi Deteksi AI

Jarak Uji (Meter)	Karakteristik Hasil	Kelebihan	Kekurangan
0.5-1	Close Up	Deteksi Wajah Sangat Akurat	Tangan/HP sering keluar frame (terpo-tong); Logika overlap gagal
1.5-3	Optimal	Seluruh Badan Akurasi COCO- SSD tinggi	
4	Jauh		Objek HP terlalu ke-cil/blur; Confidence Score turun drastis (< 50%)

Sistem bekerja optimal pada rentang jarak 1 hingga 3 meter dari kamera dengan kondisi pencahayaan cukup (minimal 300 lux). Di luar jarak tersebut, akurasi deteksi objek kecil (Handphone) menurun di bawah 50% dikarenakan keterbatasan resolusi kamera webcam standar (640x480 piksel).

Tabel 3. Kendala Teknis Berdasarkan Karakteristik Model AI

Model AI	Karakteristik Kerja	Batas Deteksi/Kinerja	Dampal Kendala
COCO - SSD (Objek)	Pengenalan Bentuk (Shape)	Orang: 5 -7 meter; HP: maks. 2 - 3 meter	HP terlihat kabur pada jarak jauh; Logika isOverlapping tidak aktif
Techable Machine (Wajah)	Pengenalan Pola Visual	Sensitif terhadap jarak latih vs uji (e.g., latih 0.5m, uji 3m)	Perubahan detail visual (dari pori-pori ke bentuk bulat) menyebabkan AI gagal

Kendala pada COCO-SSD menunjukkan bahwa resolusi input kamera sangat memengaruhi de- teksi objek kecil (HP) pada jarak jauh, sementara Teachable Machine memerlukan konsistensi jarak an- tara tahap pelatihan dan pengujian agar pola visual wajah tetap dapat dikenali secara akurat.

Tabel 4. Perbandingan Bahasa Pemrograman

Fitur	JavaScript (TensFLow.js)	Python (TensorFlow & OpenCV)
Tempat Berjalan	Browser (Chrome,Edge). Menggunakan resource RAM browser.	Sistem Operasi langsung. Mengakses hardware secara native.
Kecepatan (Performance)	Mengandalkan WebGL. Cukup cepat untuk demo, tapi terbatas oleh memori browser (limit - 4GB)	Lebih cepat dan stabil. Bisa menggunakan CUDA (NVIDIA GPU) sepenuhnya untuk performa maksimal.
Instalasi	Sangat Mudah. User cukup buka link website. Tidak perlu install apapun	Rumit. User harus install Python, pip, library, dan mengatur environtment.

Akses Hardware	Terbatas. Harus minta izin pop-up kamera. Tidak bisa akses file sistem sembarangan.	Penuh. Bisa akses kamera CCTV IP, menyimpan rekaman ke hard- disk, akses GPIO (IoT).
Privasi	Data diproses di sisi klien (aman), tapi kode sumber mudah diintip (VIew Source).	Kode Sumber bisa dicompile (.exe) sehingga logika rahasia lebih aman.
Model AI	Format JSON + BIN (Shard). Ukuran harus kecil agar cepat di-load via internet	Format H5, PB, atau ONNX. Bisa memuat model raksasa (bergiga- giga) tanpa masalah.

4. KESIMPULAN

Penelitian ini memberikan kontribusi validasi teoretis dan wawasan implementasi praktis pada domain AI object recognition berbasis web. Proses pengembangan yang terdokumentasi, dari tantangan konfigurasi Git awal hingga integrasi akhir penalaran spasial dengan platform WhatsApp, menyediakan peta jalan komprehensif untuk mengembangkan sistem keamanan inklusif. Pencapaian latensi di bawah 30 ms pada perangkat keras standar, dikombinasikan dengan pengenalan objek real-time, menunjukkan bahwa kemampuan pemantauan AI yang canggih dapat didemokratisasi melalui teknologi berbasis browser, memenuhi tujuan inti penelitian untuk membuat AI dapat diakses oleh pengguna dengan perangkat kelas menengah ke bawah.

DAFTAR PUSTAKA

[1] N. Carion, F. Massa, G. Synnaeve, N. Usunier, and A. Kirillov, "End-to-End Object Detection with Transformers," pp. 1–17, 2020.

[2] C. Wang, A. Bochkovskiy, and H. M. Liao, "YOLOv7 : Trainable bag-of-freebies sets new state- of-the-art for real-time object detectors," pp. 7464–7475, 2023.

[3] D. Smilkov *et al.*, "TensorFlow.js: Machine Learning for the Web and Beyond," *Proc. Mach. Learn. Syst.*, 2019.

[4] M. Viola, P.; Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," *Mitsubishi Electr.*, 2001.

[5] S. Ren, K. He, and R. Girshick, "Faster R-CNN : Towards Real-Time Object Detection with Re- gion Proposal Networks," pp. 1–9, 2017.

[6] W. L. B *et al.*, "SSD : Single Shot MultiBox Detector," vol. 1, pp. 21–37, 2016, doi: 10.1007/978-3-319-46448-0.

[7] K. Manji, J. Hanefeld, T. De Gruchy, J. Vearey, and H. Walls, "Using WhatsApp messenger for health systems research : a scoping review of available literature," no. April, pp. 774–789, 2021, doi: 10.1093/heapol/czab024.

[8] M. Tan, R. Pang, and Q. V Le, "EfficientDet : Scalable and Efficient Object Detection," pp. 10781–10790, 2020.

[9] C. Wang, A. Bochkovskiy, and H. M. Liao, "Scaled-YOLOv4 : Scaling Cross Stage Partial Network," pp. 13029–13038, 2021.

[10] M. Kotthapalli and D. Ravipati, "YOLOv1 to YOLOv11 : A Comprehensive Survey of Real-Time Object Detection Innovations and Challenges," pp. 1–13, 2025.

[11] D. Smilkov *et al.*, "TensorFlow.js: Machine Learning for the Web and Beyond," 2019, [Online]. Available: http://arxiv.org/abs/1901.05350

[12] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018, [Online]. Available:

http://arxiv.org/abs/1804.02767

[13] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017, [Online]. Available: http://arxiv.org/abs/1704.04861

[14] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Resid- uals and Linear Bottlenecks," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 4510–4520, 2018, doi: 10.1109/CVPR.2018.00474.

[15] S. S. S. R. Depuru, L. Wang, V. Devabhaktuni, and N. Gudi, "Smart meters for power grid - Challenges, issues, advantages and status," *2011 IEEE/PES Power Syst. Conf. Expo. PSCE 2011*, pp. 1–7, 2011, doi: 10.1109/PSCE.2011.5772451.