Integrated Journal of Applied Technology and Informatics
Journal homepage: https://journal2.uinssc.ac.id/index.php/IJATI

Fire Detection and Segmentation Using YOLOv8-Seg Based on Computer Vision

Deteksi dan Segmentasi Api Menggunakan YOLOv8-Seg Berbasis Computer Vision

Mohammad Raihan Akbar, Moch Subchi Ramadhani, Afifah Zayyin Amatillah, Muhammad Rafly Saputra, Kahfi Gunardi
Universitas Islam Negeri Siber Syekh Nurjati Cirebon, Cirebon, 45132, Indonesia
raihan13akbar@gmail.com; rsubhi33@gmail.com; afifahhhzayyin@gmail.com; Raflybiasa20@gmail.com; gunardikahfi@uinssc.ac.id;

ARTICLE INFO
Article history:
Received 2026-06-30
Revised 2026-06-30
Accepted 2026-06-30

CORRESPONDING AUTHOR
Kahfi Gunardi
gunardikahfi@uinssc.ac.id

ABSTRACT
Fire is a disaster that can cause enormous material and life losses, while conventional smoke and heat sensors respond slowly and have limited detection range. This study develops a fire detection and segmentation system based on the YOLOv8n-seg architecture, an instance segmentation model capable of producing precise pixel-level fire masks so that small/static fire objects (e.g., candles, matches, stoves) can be distinguished from large/dynamic fires (e.g., building or forest fires) to reduce false alarms. The model was trained using transfer learning on a manually annotated dataset of 240 training images and 60 validation images, with aggressive geometric and HSV color augmentation to address the limited dataset size. Training was performed for 50 epochs on a Google Colab NVIDIA Tesla T4 GPU. Evaluation results show a mask segmentation precision of 98.0%, recall of 96.4%, and mAP@50 of 98.2%, with per-class AP@50 of 99.50% for small/static fire and 96.92% for large/dynamic fire. The exported model (6.45 MB) achieved a real-time inference speed of approximately 30.4 ms per frame (~33 FPS) when tested on a live webcam stream, indicating its feasibility for real-time fire early-warning applications.
Keyword: Fire Detection, YOLOv8-Seg, Instance Segmentation, Computer Vision, Real-Time

ABSTRAK
Kebakaran merupakan bencana yang berpotensi menimbulkan kerugian material dan korban jiwa yang besar, sementara sensor asap dan panas konvensional memiliki keterbatasan jarak dan waktu respon. Penelitian ini mengembangkan sistem deteksi dan segmentasi api berbasis arsitektur *YOLOv8n-seg*, sebuah model segmentasi instansi yang mampu menghasilkan *masker* piksel api secara presisi sehingga objek api kecil/statis (seperti lilin, korek api, kompor) dapat dibedakan dari api besar/dinamis (seperti kebakaran gedung atau hutan) guna meminimalkan alarm palsu. Model dilatih menggunakan *transfer learning* pada dataset hasil anotasi manual sebanyak 240 gambar latih dan 60 gambar validasi, dengan augmentasi geometris dan warna (HSV) yang agresif untuk mengatasi keterbatasan ukuran dataset. Pelatihan dilakukan selama 50 *epoch* menggunakan *GPU NVIDIA Tesla T4* pada *Google Colab*. Hasil evaluasi menunjukkan *precision* segmentasi *masker* sebesar 98,0%, *recall* 96,4%, dan *mAP*@50 mencapai 98,2%, dengan *AP*@50 per kelas sebesar 99,50% untuk api kecil/statis dan 96,92% untuk api besar/dinamis. Model yang diekspor (6,45 MB) mencapai kecepatan inferensi *real-time* sekitar 30,4 ms per *frame* (~33 FPS) saat diuji pada aliran video *webcam* langsung, menunjukkan kelayakannya untuk diterapkan sebagai sistem peringatan dini kebakaran secara nyata.
Keyword: Deteksi Api, YOLOv8-Seg, Segmentasi Instansi, Computer Vision, Real-Time

1. PENDAHULUAN

Kebakaran merupakan salah satu bencana yang dapat menyebabkan kerugian material, kerusakan infrastruktur, pencemaran lingkungan, hingga korban jiwa. Kecepatan dalam mendeteksi munculnya api menjadi faktor penting untuk meminimalkan dampak yang ditimbulkan melalui tindakan mitigasi dan penanganan secara dini. Oleh karena itu, sistem deteksi kebakaran yang mampu memberikan respons secara cepat dan akurat menjadi kebutuhan penting pada berbagai lingkungan, seperti gedung perkantoran, kawasan industri, gudang logistik, maupun kawasan hutan. Penelitian terkini menunjukkan bahwa sistem deteksi kebakaran berbasis kecerdasan buatan mampu meningkatkan kecepatan deteksi sekaligus mengurangi *false alarm* dibandingkan pendekatan konvensional [1], [2]. Sistem deteksi kebakaran konvensional umumnya memanfaatkan *smoke detector* dan *heat detector* untuk mendeteksi keberadaan asap atau peningkatan suhu di sekitar sensor. Meskipun telah digunakan secara luas, pendekatan tersebut memiliki beberapa keterbatasan, terutama pada area dengan cakupan yang luas, bangunan bertingkat dengan langit-langit tinggi, maupun lingkungan terbuka. Sensor hanya akan memberikan respons setelah asap atau panas mencapai lokasi sensor sehingga waktu deteksi dapat menjadi lebih lambat. Selain itu, kondisi lingkungan tertentu juga berpotensi meningkatkan terjadinya alarm palsu (*false alarm*). Perkembangan teknologi *Computer Vision* memberikan alternatif melalui pemanfaatan kamera pengawas (CCTV) yang mampu melakukan pemantauan secara kontinu dari jarak jauh tanpa memerlukan pemasangan sensor tambahan pada setiap lokasi pengamatan [1], [3].

Computer Vision merupakan cabang ilmu komputer yang berfokus pada pengembangan metode agar komputer mampu memperoleh, memahami, dan menginterpretasikan informasi dari citra maupun video digital secara otomatis [4]. Dalam beberapa tahun terakhir, perkembangan *Deep Learning*, khususnya *Convolutional Neural Network* (CNN), telah meningkatkan kemampuan sistem *Computer Vision* dalam berbagai tugas seperti klasifikasi citra, deteksi objek, hingga segmentasi citra dengan tingkat akurasi yang semakin tinggi [5]. Berbeda dengan deteksi objek yang hanya menghasilkan *bounding box*, segmentasi citra mampu melakukan klasifikasi pada tingkat piksel sehingga bentuk objek dapat direpresentasikan secara lebih rinci. Kemampuan tersebut menjadi penting pada objek api yang memiliki bentuk tidak beraturan dan selalu berubah mengikuti kondisi lingkungan [6], [7]. Salah satu algoritma yang banyak digunakan pada aplikasi deteksi objek secara *real-time* adalah keluarga *You Only Look Once* (YOLO). Generasi terbaru, yaitu *YOLOv8*, mengadopsi pendekatan *anchor-free* sehingga mampu meningkatkan efisiensi deteksi dengan tetap mempertahankan kecepatan inferensi yang tinggi [8]. Selain mendukung deteksi objek, YOLOv8 juga menyediakan model *instance segmentation* (YOLOv8-seg) yang mampu menghasilkan *bounding box* dan *masker* segmentasi secara bersamaan [9]. Varian *YOLOv8n-seg* dirancang sebagai model ringan dengan jumlah parameter yang relatif kecil sehingga sesuai untuk implementasi pada perangkat dengan sumber daya komputasi terbatas maupun aplikasi *real-time* [9], [10].

Berbagai penelitian telah memanfaatkan algoritma berbasis *Deep Learning* untuk mendeteksi api maupun asap. Muhammad et al. mengembangkan sistem deteksi api berbasis *Convolutional Neural Network* pada video pengawasan dan menunjukkan bahwa pendekatan *deep learning* mampu meningkatkan akurasi deteksi dibandingkan metode berbasis warna [11]. Talaat mengembangkan model deteksi api menggunakan *YOLOv8* untuk lingkungan *smart city* dan memperoleh peningkatan performa dibandingkan pendekatan sebelumnya [1]. Selain itu, Cao et al. mengembangkan metode *YOLO-SF* untuk segmentasi api yang mampu menghasilkan representasi objek secara lebih presisi dibandingkan deteksi menggunakan *bounding box* saja [12]. Meskipun demikian, sebagian besar penelitian tersebut masih berfokus pada deteksi keberadaan api atau segmentasi objek tanpa membedakan karakteristik api berdasarkan tingkat kebakaran.

Berdasarkan kondisi tersebut, penelitian ini mengembangkan sistem deteksi dan segmentasi api menggunakan arsitektur *YOLOv8n-seg* dengan memanfaatkan pendekatan *instance segmentation*. Penelitian ini mengklasifikasikan objek menjadi dua kategori, yaitu api kecil/statis dan api besar/dinamis, sehingga selain mampu mendeteksi keberadaan api, sistem juga dapat memberikan informasi mengenai kategori api yang terdeteksi. Model dilatih menggunakan dataset yang telah dianotasi dalam format segmentasi poligon *YOLO* dan dievaluasi menggunakan metrik *Precision*, *Recall*, *mAP@50*, *mAP@50–95*, serta kecepatan inferensi (*Frame Per Second*/FPS) untuk mengetahui kelayakan model pada implementasi *real-time*.

2. METODE

2.1. Dataset Penelitian

Dataset utama yang digunakan dalam penelitian ini disimpan pada *Google Drive* dengan nama folder 'dataset_gabungan_api_seg', yang telah dibagi ke dalam *subset training* dan *validation*. Distribusi data yang digunakan disajikan pada Tabel 1.

Tabel 1. Distribusi dataset penelitian.

Subset Dataset	Jumlah Gambar	Jumlah Label Anotasi
Training Set (Data Latih)	240	297
Validation Set (Data Uji)	60	75

Dataset di atas dianotasi secara manual dengan format poligon segmentasi *YOLO*. Kelas objek yang digunakan didefinisikan sebagai berikut: Kelas 0 = Api kecil/Statis (representasi visual api lilin, korek api, kompor gas skala rumahan); Kelas 1 = Api besar/Dinamis (representasi visual kebakaran hutan, gedung, kobaran api besar tak terkontrol).

2.2. Diagram Alur Sistem

Sistem pemrosesan dari awal hingga pengoperasian nyata mengikuti alur berikut: *Dataset API* (*Google Drive*) → *Resize* (640×640 piksel) & *Augmentasi HSV*/Rotasi → *Training YOLOv8n-seg* (*Google Colab*) → Validasi & Metrik (menghasilkan best.pt) → *Inferensi Live Stream* (camera.py).

2.3. Preprocessing dan Augmentasi Data

Gambar masukan di-*resize* secara otomatis ke ukuran 640×640 piksel. Guna memperkaya variasi data dan menghindari *overfitting* akibat ukuran dataset yang kecil, diterapkan parameter augmentasi khusus pada fungsi latih *YOLOv8-seg* sebagaimana disajikan pada Tabel 2.

Tabel 2. Parameter augmentasi data pelatihan.

Augmentasi	Nilai Parameter	Deskripsi Pengaruh Model
degrees (Rotasi)	15.0	Memutar gambar acak hingga 15° untuk simulasi api miring ditiup angin.
translate (Pergeseran)	0.1	Menggeser posisi objek api secara horizontal/vertikal untuk melatih deteksi di sudut gambar.
scale (Penskalaan)	0.2	Mengubah ukuran skala objek sebesar ±20% guna simulasi jarak api yang menjauh/mendekat.
fliplr (Pencerminan)	0.5	Pencerminan horizontal gambar dengan probabilitas 50% untuk memperkaya sudut orientasi.
hsv_h (Hue Jitter)	0.015	Mengubah warna dasar (hue) api secara acak dari merah pekat, jingga, hingga kuning terang.
hsv_s (Saturation)	0.3	Mengubah saturasi warna api untuk menyimulasikan api yang sangat pekat atau memudar.
hsv_v (Value/Brightness)	0.3	Mengubah kecerahan gambar untuk mensimulasikan pencahayaan siang hari terik atau malam hari.

2.4. Arsitektur dan Parameter Training Model

Model dilatih menggunakan *transfer learning* dengan memuat bobot awal *pretrained yolov8n-seg.pt*. *YOLOv8-seg* bekerja dengan membagi jaringan menjadi dua cabang keluaran utama secara *paralel*, yaitu *proto-mask branch* yang

menghasilkan sejumlah *masker prototipe* resolusi tinggi, *detection* & *coefficient branch* yang memprediksi koordinat *bounding box*, klasifikasi kelas, serta koefisien *masker* untuk setiap objek terdeteksi. Hasil *masker* akhir diperoleh dari perkalian matriks linear antara *masker* prototipe dengan koefisien *masker* yang sesuai. *YOLOv8n-seg* dipilih karena merupakan varian model terkecil (±3,2 juta parameter) sehingga hemat daya komputasi dan ideal untuk pengoperasian *real-time*. Parameter-parameter penting yang dikonfigurasi saat proses pelatihan disajikan pada Tabel 3.

Tabel 3. Parameter pelatihan model YOLOv8n-seg.

Parameter Latih	Nilai Konfigurasi	Tujuan & Alasan Pemilihan
Epochs	50	Menentukan 50 kali perulangan pelatihan seluruh dataset gambar.
Image Size (imgsz)	640	Ukuran resolusi citra input model.
Batch Size	16	Jumlah gambar yang dimuat dalam memori GPU per iterasi update bobot.
Optimizer	AdamW (Auto)	Pilihan optimasi laju pembelajaran adaptif yang stabil.
Patience (Early Stop)	15	Menghentikan latihan jika mAP tidak naik selama 15 epoch berturut-turut.

2.5. Implementasi Perangkat Lunak dan Perangkat Keras

Implementasi kode program dan pelatihan model menggunakan ekosistem *Python* dengan detail spesifikasi teknologi sebagaimana disajikan pada Tabel 4.

Tabel 4. Spesifikasi perangkat lunak dan perangkat keras.

Kategori	Nama Teknologi	Spesifikasi / Fungsi
Bahasa Pemrograman	Python v3.12.13	Digunakan sebagai bahasa utama pemrograman.
Library Deep Learning	Ultralytics v8.4.63 & PyTorch v2.11.0	Kerangka kerja utama model YOLOv8-seg.
Library Pengolahan Citra	OpenCV-Python	Menangani operasi input video dan render visual webcam.
Akselerasi Perangkat Keras	GPU NVIDIA Tesla T4 (Cloud)	Memiliki 15 GB GDDR6 VRAM untuk akselerasi pelatihan.
Sistem Operasi	Ubuntu Linux (Cloud) / Windows (Lokal)	Lingkungan eksekusi latihan dan pengujian program.

3. HASIL DAN PEMBAHASAN

3.1. Hasil Training Model

Proses pelatihan model *YOLOv8n-seg* berjalan lancar selama 50 *epoch*. Penurunan fungsi kerugian (*loss*) pada data latih dan validasi terpantau konsisten, membuktikan bahwa model mampu mempelajari fitur segmentasi api secara progresif. Rincian perkembangan nilai *loss* dirangkum pada Tabel 5.

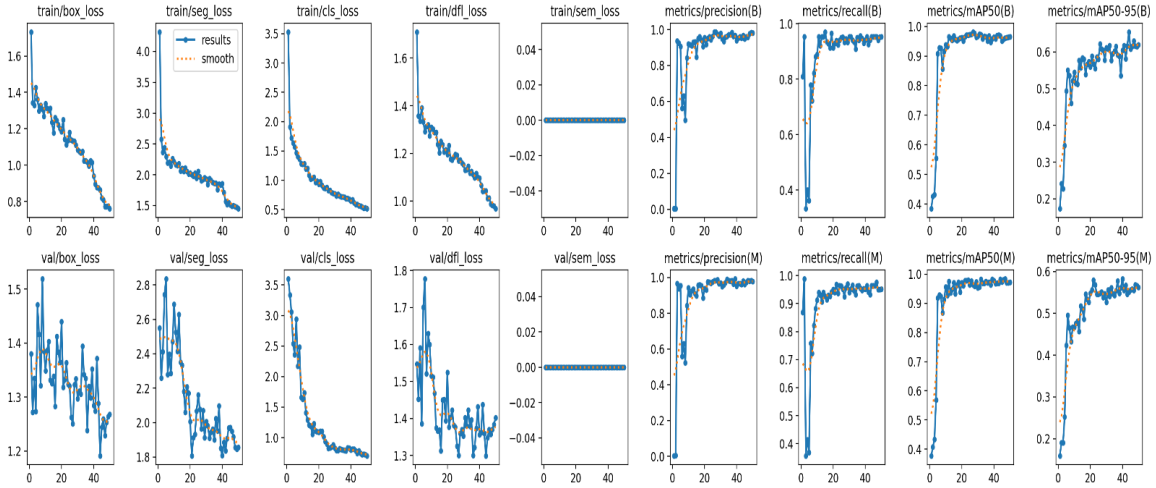
Tabel 5. Perkembangan nilai loss selama pelatihan

Epoch	Box Loss	Seg Loss	Class Loss	DFL Loss
1 / 50	1.7330	4.3190	3.5320	1.7100
10 / 50	1.3370	2.2120	1.2780	1.3100
30 / 50	1.0850	1.9340	0.7670	1.1530
50 / 50 (Akhir)	0.7607	1.4500	0.5158	0.9685

Penurunan *Box Loss* (koordinat *bounding box*) dan *Seg Loss* (akurasi *masker* piksel) menunjukkan peningkatan presisi spasial dari model. *Class Loss* yang menurun tajam hingga mencapai nilai 0,5158 membuktikan kemampuan klasifikasi model yang sangat mantap dalam mengidentifikasi kelas Api Kecil/Statis dan Api Besar/Dinamis.

3.2. Pengujian Gambar Validasi (Batch Prediksi)

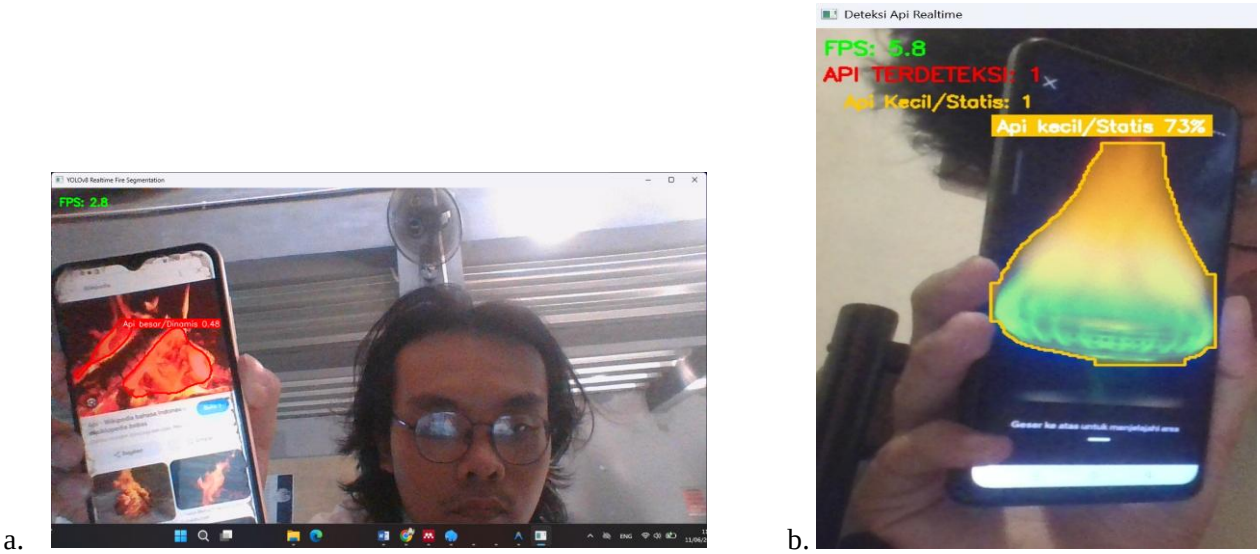
Model terbaik (best.pt) diuji secara acak terhadap 10 gambar dari dataset validasi untuk melihat hasil prediksi visualnya. Hasil pengujian menunjukkan seluruh objek api terdeteksi dan tersegmentasi dengan koordinat poligon yang sangat rapat, sebagaimana ditunjukkan pada Gambar 1.



Gambar 1. Hasil prediksi segmentasi model pada batch gambar validasi.

3.3. Pengujian Video Real-Time via Webcam

Model yang diekspor (best.pt) dengan ukuran 6,45 MB berhasil diintegrasikan ke dalam skrip *Python* camera.py. Pada pengujian menggunakan kamera lokal, skrip berhasil membaca aliran video, melakukan inferensi model dengan ambang batas (*confidence threshold*) sebesar 0,25, menggambar *overlay masker* warna semitransparan (kuning-jingga untuk api kecil, merah untuk api besar), serta menampilkan indikator keamanan dan FPS secara dinamis pada layar, sebagaimana ditunjukkan pada Gambar 2.



a. b.

Gambar 2. (a) Hasil *real-time* model pada foto *webcam*; (b) Hasil *real-time* model pada video *webcam*;

3.4. Evaluasi Metrik Performa Model

Pengujian formal menggunakan metrik evaluasi standar *computer vision* pada subset validasi (60 gambar, 63 instance objek api) memberikan hasil yang sangat memuaskan, dengan akurasi segmentasi *masker mAP@50* mencapai 98,2%. Perbandingan metrik performa disajikan pada Tabel 6.

Tabel 6. Perbandingan metrik performa deteksi *bounding box* dan segmentasi *masker*

Tipe Deteksi	Precision (P)	Recall (R)	mAP@50	mAP@50-95

Bounding Box	0.9680 (96.8%)	0.9520 (95.2%)	0.9690 (96.9%)	0.6520 (65.2%)
Segmentation (Mask)	0.9800 (98.0%)	0.9640 (96.4%)	0.9821 (98.2%)	0.5524 (55.2%)

Metrik deteksi *masker* piksel per kelas (*AP@50*) disajikan pada Tabel 7.

Tabel 7. *Average Precision* (AP@50) *masker* per kelas

Nama Kelas Target	**Average Precision (AP@50) – *Masker***
Api kecil/Statis	0.9950 (99.50%)
Api besar/Dinamis	0.9692 (96.92%)

Model ini juga memiliki waktu respons yang sangat cepat. Rincian waktu inferensi per gambar adalah: *preprocessing* 6,6 ms, *inference* 16,1 ms, *postprocessing* 7,7 ms, dengan total waktu komputasi 30,4 ms (setara ≈33 FPS). Kecepatan ini membuktikan kelayakan model untuk dioperasikan pada sistem tanggap darurat waktu-nyata.

3.5. Analisis Kelebihan dan Kelemahan Sistem

Sistem ini memiliki keunggulan utama pada ukuran model yang sangat kecil (6,45 MB) dan kecepatan pemrosesan *frame rate* tinggi (≈33 FPS). Selain itu, akurasi segmentasi *masker mAP@50* yang mencapai 98,2% membuktikan bahwa model mampu melakukan pemetaan piksel batas tepi api dengan keandalan yang sangat tinggi. Sistem ini juga mampu memisahkan kelas api kecil dan api besar secara presisi, yang berguna meminimalkan alarm palsu.

Kelemahan utama terletak pada ukuran dataset latih yang relatif kecil (240 gambar), sehingga model rentan terhadap *false positive* jika dihadapkan pada objek bercahaya jingga/merah statis (seperti lampu halogen atau papan reklame neon) apabila ditempatkan di lingkungan luar ruangan dengan pencahayaan ekstrem.

3.6. Kendala dan Solusi Selama Proyek

Beberapa hambatan teknis yang dihadapi selama pengerjaan proyek beserta solusi penyelesaiannya dirangkum pada Tabel 8.

Tabel 8. Kendala teknis dan solusi penyelesaian selama proyek.

Aspek Hambatan	**Kendala Teknis**	**Solusi Penyelesaian**
Komputasi Lokal	Komputer lokal tidak didukung oleh GPU CUDA berkapasitas besar, menyebabkan waktu training model deep learning sangat lama.	Memanfaatkan Google Colab yang menyediakan GPU NVIDIA Tesla T4 15 GB secara gratis, memotong durasi training menjadi ±6,7 menit.
Variabilitas Api	Bentuk kobaran api sangat dinamis dan mudah terdistorsi oleh tiupan angin, menyulitkan lokalisasi poligon.	Menerapkan konfigurasi augmentasi spasial dan warna yang agresif pada parameter model.train() (rotasi, translasi, flip, dan pergeseran HSV).

4. KESIMPULAN

Berdasarkan seluruh hasil perancangan, pelatihan, pengujian, dan analisis evaluasi model segmentasi api, dapat disimpulkan bahwa model *deep learning* deteksi dan segmentasi instansi api berhasil dilatih menggunakan arsitektur *YOLOv8n-seg* dengan ukuran model akhir yang sangat kompak (6,45 MB). Proses latih model selama 50 *epoch* dengan *transfer learning* dan augmentasi data khusus api menghasilkan tingkat presisi segmentasi *masker* sebesar 98,0%, *recall* 96,4%, dan *metrik mAP@50* mencapai 98,2%. Sistem ini terbukti andal dalam memisahkan dan mengklasifikasikan kelas Api Kecil/Statis (*AP@50* = 99,50%) serta kelas Api Besar/Dinamis (*AP@50* = 96,92%). Pengujian *real-time* dengan *webcam* melalui skrip camera.py menunjukkan respons deteksi yang cepat dengan kecepatan inferensi total ≈30,4 ms (≈33 FPS), sehingga sangat layak untuk diimplementasikan secara langsung di lapangan.

Untuk penyempurnaan sistem pemantauan kebakaran di masa mendatang, pengembangan lanjutan disarankan pada beberapa aspek, yaitu menambah volume dataset latih dengan mengintegrasikan gambar-gambar kondisi berasap (*smoke detection*) dan variasi latar belakang malam hari luar ruangan; mengintegrasikan program camera.py dengan sistem notifikasi peringatan otomatis (seperti Telegram API Bot atau SMTP email) untuk mengirim tangkapan layar api secara *instan* ke tim keamanan gedung; melakukan *porting* dan pengujian model best.pt pada perangkat *edge computing* hemat

daya seperti *Raspberry Pi 5* atau *NVIDIA Jetson Nano*; serta mengeksplorasi penggunaan arsitektur model segmentasi yang lebih besar (seperti *YOLOv8s-seg* atau *YOLOv8m-seg*) apabila sistem dijalankan pada *server* terpusat dengan *GPU CUDA* khusus guna mendongkrak skor *mAP@50-95*.

DAFTAR PUSTAKA

[1] F. M. Talaat and H. ZainEldin, "An Improved Fire Detection Approach Based on YOLO-v8 for Smart Cities," *Neural Computing and Applications*, vol. 35, no. 28, pp. 20939–20954, 2023, doi: 10.1007/s00521-023-08809-1. Available: https://doi.org/10.1007/s00521-023-08809-1

[2] B. M. Diaconu, "Recent Advances and Emerging Directions in Fire Detection Systems Based on Machine Learning Algorithms," *Fire*, vol. 6, no. 11, Art. no. 441, 2023, doi: 10.3390/fire6110441.

[3] S. Verstockt, B. Merci, F. Sette, B. Lambert, and R. Van de Walle, "State of the Art in Vision-Based Fire and Smoke Detection," *Fire Safety Journal*, vol. 83, pp. 24–37, 2016.

[4] R. Szeliski, *Computer Vision: Algorithms and Applications*, 2nd ed. Cham, Switzerland: Springer, 2022.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017, pp. 2961–2969.

[7] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 2015, pp. 3431–3440.

[8] G. Jocher, A. Chaurasia, and Ultralytics Team, *Ultralytics YOLO Documentation*. Available: https://docs.ultralytics.com/

[9] G. Jocher *et al.*, *Ultralytics YOLOv8*. GitHub Repository. Available: https://github.com/ultralytics/ultralytics

[10] S. Shorten and T. M. Khoshgoftaar, "A Survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 60, 2019, doi: 10.1186/s40537-019-0197-0.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 779–788.

[12] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010, doi: 10.1007/s11263-009-0275-4.