Jurnal ICT : Information Communication & Technology
Vol. 25, N0.2, Desemmber 2025, pp. 154-158
p-ISSN: 2302-0261, e-ISSN: 2303-3363, DOI: https://doi.org/10.36054/jict-ikmi.v25i2.330

Fire Detection Berbasis *Computer Vision* Menggunakan YOLOv8 Secara *Real-Time*

Rizq Muhammad Sukmosuwarno[1], Muhammad Faris Fadhil Islam[2], Raden Muhammad Raditya Rahman[3], Gema Parasti Mindara[4], Endang Purnama Giri[5]
[1-3]Program Studi Teknologi Rekayasa Perangkat Lunak, Sekolah Vokasi IPB University, Bogor, Indonesia
[4]Program Studi Teknologi Rekayasa Komputer, Sekolah Vokasi IPB University, Bogor, Indonesia
[5]Program Studi Ilmu Komputer, Sekolah Sains, Data, Matematika, dan Informatika, IPB University, Bogor, Indonesia
Email: [1]ais_muhammadsukmosuwarno@apps.ipb.ac.id, [2]radttyaarahman@apps.ipb.ac.id, [3]farisfadhil_islam@apps.ipb.ac.id, [4]gemaparasti@apps.ipb.ac.id, [5]endang_pg@apps.ipb.ac.id

INFORMASI ARTIKEL

Histori artikel:

Naskah masuk, 30 November 2025
Direvisi, 15 Desember 2025
Diiterima, 23 Desember 2025

Kata Kunci:

YOLOv8,
Real-Time,
OpenCV,
Pengolahan Citra,
Deteksi Api.

ABSTRAK

Abstract- *This study developed an image processing-based fire detection system using the YOLOv8 algorithm to support real-time early fire detection. The dataset used consisted of 100 fire images with varying visual conditions, which were processed through pre-processing, annotation conversion to YOLO format, and model training for 30 epochs. Performance evaluation shows that the YOLOv8 model achieves mAP50 of 0.646, precision of 0.889, and recall of 0.455, with an inference speed of 282.5 ms/frame. The system is integrated with OpenCV to process webcam input and display detection results directly. Although the results show the potential of YOLOv8 in real-time fire detection, this study has limitations in the relatively small amount of data, so model generalization still needs to be improved. Further research is recommended using a larger and more diverse dataset and comparing it with other detection models.*

Abstrak- Penelitian ini mengembangkan sistem deteksi api berbasis pengolahan citra menggunakan algoritma YOLOv8 untuk mendukung deteksi dini kebakaran secara real-time. Dataset yang digunakan terdiri dari 100 citra api dengan variasi kondisi visual, yang diproses melalui tahap pra-pemrosesan, konversi anotasi ke format YOLO, serta pelatihan model selama 30 epoch. Evaluasi kinerja menunjukkan bahwa model YOLOv8 mencapai mAP50 sebesar 0.646, precision 0.889, dan recall 0.455, dengan kecepatan inferensi 282.5 ms/frame. Sistem diintegrasikan dengan OpenCV untuk memproses input webcam dan menampilkan hasil deteksi secara langsung. Meskipun hasil menunjukkan potensi YOLOv8 dalam deteksi api real-time, penelitian ini memiliki keterbatasan pada jumlah dataset yang relatif kecil, sehingga generalisasi model masih perlu ditingkatkan. Penelitian selanjutnya disarankan menggunakan dataset yang lebih besar dan beragam serta melakukan perbandingan dengan model deteksi lainnya.

***Penulis Korespondensi**:*

Gema Parasti Mindara
Program Studi Teknologi Rekayasa Komputer,
IPB University
Jl. Kumbang, Kecamatan Bogor Tengah, Kota Bogor, Jawa Barat
Email: gemaparasti@apps.ipb.ac.id

1. Pendahuluan

Deteksi api secara cepat dan akurat menjadi faktor penting dalam upaya mitigasi bencana. Sistem deteksi konvensional seperti sensor asap dan sensor panas masih memiliki berbagai keterbatasan, antara lain keterlambatan deteksi, sensitivitas terhadap kondisi lingkungan, serta area cakupan yang relatif sempit [1].

Perkembangan teknologi *computer vision* yang didukung oleh kemajuan *deep learning* membuka peluang baru dalam sistem deteksi kebakaran berbasis citra. salah satunya *Deep Residual Learning* yang diperkenalkan oleh He et al. [2], yang memungkinkan ekstraksi fitur visual secara lebih mendalam dan efektif pada model modern.

Salah satu metode yang paling populer dalam *computer vision* adalah *You Only Look Once (YOLO)*. YOLO dirancang untuk melakukan deteksi objek secara end-to-end dalam satu tahap pemrosesan, sehingga mampu mencapai kecepatan inferensi yang tinggi. versi terbarunya, YOLOv8, menawarkan peningkatan akurasi, arsitektur yang lebih efisien, serta dukungan penuh untuk implementasi *real-time* [3], [4]. Dengan dukungan dataset beranotasi seperti PASCAL VOC [5], sistem berbasis YOLO mampu mendeteksi objek dalam berbagai kondisi pencahayaan, sudut pandang, dan ukuran objek.

Meskipun berbagai penelitian sebelumnya telah mengkaji deteksi api menggunakan model *deep learning*, sebagian besar studi berfokus pada evaluasi menggunakan dataset berskala besar atau dilakukan secara *offline*. Selain itu, kajian yang secara khusus mengevaluasi kinerja YOLOv8 untuk deteksi api pada skenario *real-time* dengan dataset berskala terbatas masih relatif terbatas. Kondisi ini menunjukkan adanya *research gap* terkait analisis performa YOLOv8 dalam konteks keterbatasan data serta implementasi langsung pada sistem pemantauan berbasis kamera.

Berdasarkan latar belakang tersebut, penelitian ini bertujuan untuk mengembangkan dan mengevaluasi sistem deteksi api berbasis YOLOv8 menggunakan dataset citra beranotasi format Pascal VOC yang dikonversi ke format YOLO. Model dilatih selama 30 epoch dan diimplementasikan pada sistem deteksi api *real-time* menggunakan kamera berbasis OpenCV. Penelitian ini diharapkan dapat memberikan gambaran kinerja YOLOv8 pada skenario *real-time* serta mengidentifikasi tantangan dan keterbatasan yang muncul akibat penggunaan dataset berskala kecil.

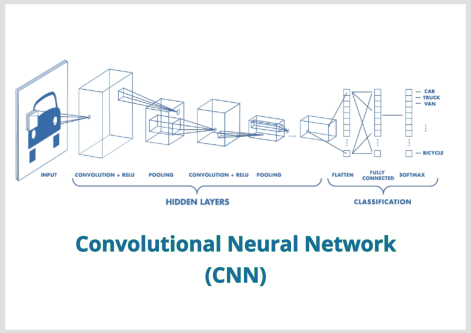
2. Tinjauan Pusataka

2.1 Pengolahan Citra Digital

Pengolahan citra digital adalah proses memodifikasi, meningkatkan, atau mengekstraksi informasi dari gambar digital. Teknik ini banyak diterapkan untuk segmentasi, deteksi objek, klasifikasi, dan kompresi [6]. Dalam penelitian ini, pengolahan citra digunakan untuk menganalisis gambar api secara otomatis.

2.2 Deep Learning

Deep Learning merupakan cabang dari *machine learning* yang meniru kerja otak dengan menggunakan jaringan saraf tiruan [7]. Model *deep learning* modern seperti YOLOv8 menggunakan *convolutional layers* untuk mengekstraksi fitur dari citra.



Gambar 1. Metode Convolution Neural Network

Gambar 1 menunjukkan alur kerja dasar dari *Convolutional Neural Network (CNN)* yang digunakan untuk memproses citra. Model CNN terdiri dari beberapa lapisan utama, yaitu *convolutional layer, pooling layer, dan fully connected layer*. Pada tahap awal, citra dimasukkan ke convolutional layer untuk mengekstraksi fitur penting seperti tepi, tekstur, dan pola warna khas api.

2.2.1 **Pengumpulan dan Persiapan Dataset**

Tahap pertama melibatkan pengumpulan dataset citra api dan non-api yang diperoleh dari sumber terbuka dan dokumentasi proses pengambilan gambar secara langsung di lokasi nyata. Citra kemudian diproses melalui tahapan pra pemrosesan seperti *resize*, normalisasi piksel, serta *data augmentation* untuk memperluas variasi dataset.

2.2.1 **Ekstraksi Fitur Menggunakan Convolutional Layer**

Pada tahap ini, citra yang telah di praproses dimasukkan ke dalam beberapa *convolutional layer* untuk mengekstraksi fitur penting. Setiap lapisan konvolusi menggunakan sejumlah filter untuk mendeteksi pola seperti tepi, warna khas api, dan tekstur nyala.

2.2.3 **Reduksi Dimensi dengan Pooling Layer**

Setelah fitur diekstraksi, CNN melakukan proses *pooling* untuk mereduksi ukuran fitur sekaligus mempertahankan informasi utama. Penelitian ini menggunakan *max pooling* karena mampu mengambil fitur dominan yang umumnya muncul pada citra api.

2.2.4 **Mekanisme Klasifikasi dan Deteksi pada YOLOv8**

Pada arsitektur YOLOv8, proses klasifikasi tidak dilakukan menggunakan *fully connected layer* sebagaimana pada Convolutional Neural Network (CNN) konvensional. Fitur hasil ekstraksi dari backbone CNN diproses secara end-to-end dan diteruskan ke *detection head* YOLOv8 untuk

melakukan prediksi kelas objek dan koordinat *bounding box* secara langsung. Pendekatan ini memungkinkan YOLOv8 melakukan deteksi objek berbasis *anchor-free*, sehingga proses klasifikasi dan lokalisasi objek dilakukan secara simultan tanpa tahap *flattening* dan *fully connected layer* terpisah. Dengan mekanisme ini, YOLOv8 mampu mencapai kecepatan inferensi yang tinggi dan performa yang efisien untuk aplikasi deteksi api secara real-time.

2.2.5 Evaluasi dan Optimasi Model

Tahap berikutnya adalah evaluasi performa model menggunakan metrik akurasi, presisi, recall, dan *F1-score*. Evaluasi dilakukan menggunakan data uji yang dipisahkan dari data pelatihan untuk memastikan objektivitas nilai kinerja. Ketika model belum optimal, dilakukan *hyperparameter tuning*, penyesuaian arsitektur CNN, atau penambahan *augmentation*.

2.3 YOLOv8

YOLOv8 merupakan versi terbaru dari YOLO yang dikembangkan oleh Ultralytics. YOLOv8 memiliki peningkatan struktur model, mendukung *backbone* CSPDarknet modern, *anchor-free detection*, serta performa yang lebih cepat dibanding pendahulunya [8], [9]. YOLOv8 mendukung berbagai ukuran model seperti YOLOv8n (*nano*), YOLOv8s (*small*), hingga YOLOv8x (*extra-large*). Pada penelitian ini digunakan YOLOv8 yang lebih ringan dan cepat untuk *real-time.*

2.4 Deteksi Api Berbasis Visi Komputer

Deteksi api berbasis citra memiliki beberapa keunggulan seperti jangkauan luas, tidak mudah salah deteksi akibat perubahan suhu, dan dapat bekerja di luar ruangan. Berbagai metode telah digunakan, mulai dari segmentasi warna hingga *deep learning* modern seperti YOLO dan EfficientDet [10].

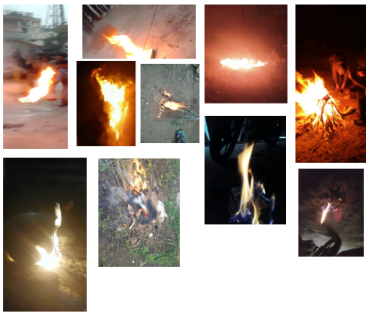
3. Metode Penelitian

3.1 *Dataset*

Dataset yang digunakan dalam penelitian ini terdiri dari 100 citra api yang diperoleh dari berbagai sumber terbuka. Meskipun jumlah dataset relatif terbatas untuk pelatihan model deteksi objek berbasis *deep learning* seperti YOLOv8, dataset ini tetap digunakan untuk mengevaluasi kemampuan model dalam skenario *real-time* dengan keterbatasan data. Citra yang digunakan memiliki variasi visual yang mencakup perbedaan ukuran nyala api, intensitas cahaya, latar belakang, serta sudut pengambilan gambar, sehingga tetap mempresentasikan kondisi lingkungan nyata secara beragam.

Dataset dibagi menjadi dua bagian, yaitu 80% sebagai data latih (training set) dan 20% sebagai data uji (testing set). Pembagian ini dilakukan untuk memastikan bahwa evaluasi kinerja model dilakukan secara objektif menggunakan data yang tidak terlibat dalam proses pelatihan.

Visualisasi contoh citra dalam *dataset* dapat dilihat pada Gambar 2 berikut:



Gambar 2. Contoh variasi visual gambar api

3.2 Augmentasi Data

Untuk mengatasi keterbatasan jumlah dataset dan meningkatkan kemampuan generalisasi model, dilakukan teknik *data augmentation*. Teknik augmentasi yang digunakan meliputi rotasi citra hingga .±15°, horizontal flipping, penyesuaian tingkat kecerahan dan kontras, serta scaling. Augmentasi ini bertujuan untuk memperkaya variasi visual data latih sehingga model lebih robust terhadap perubahan kondisi pencahayaan dan sudut pandang pada implementasi *real-time*.

3.3 Desain Sistem

Penelitian ini menggunakan metode deteksi objek berbasis YOLOv8 untuk mendeteksi api secara *real-time.* Sistem dirancang agar mampu membaca *input* video dari webcam, kemudian memproses setiap *frame* menggunakan model yang telah dilatih. Arsitektur YOLOv8 dipilih karena memiliki kemampuan deteksi objek yang cepat dan akurat, sesuai dengan kebutuhan aplikasi pemantauan kebakaran [1], [3].

3.4 Arsitektur YOLOv8

YOLOv8 memiliki tiga komponen utama:

- **Backbone -** mengekstraksi fitur
- **Neck** - menggabungkan fitur multilevel
- **Head** - memprediksi bounding box dan label

Arsitektur ini cepat dan akurat sehingga cocok untuk deteksi real-time [2], [5].

3.5 Pelatihan Model

Setelah pelatihan selesai, model dievaluasi menggunakan data pengujian untuk mengukur performa secara objektif. Metrik evaluasi yang digunakan meliputi akurasi, *precision*, *recall*, dan

F1-score. Hasil pengujian menunjukkan bahwa model mampu mendeteksi keberadaan api dengan tingkat akurasi tinggi.

Table 1. Hasil Parameter Model

Parameter	Nilai
Epoch	30
Learning rate	0.001
Batch Size	16
Ukuran Gambar	640x640
Model Dasar	YOLOv8

3.6 Implementasi Sistem *Real-Time*

Model YOLOv8 digabungkan dengan OpenCV untuk deteksi webcam. Sistem akan menampilkan *bounding box & confidence score.* Implementasi *real-time* ini cocok digunakan untuk sistem monitoring kebakaran rumah tangga maupun industri [7].

4. Hasil dan Pembahasan

4.1 Hasil Pelatihan Model YOLOv8

Proses pelatihan model YOLOv8 dilakukan selama 30 *epoch* menggunakan konfigurasi seperti ukuran citra 640×640, batch size 16, serta learning rate 0.001. Tujuan pelatihan ini adalah agar model mampu mendeteksi objek api dengan akurasi tinggi dan mampu bekerja secara real-time [1], [2].

Model menunjukkan peningkatan performa secara bertahap selama proses training, yang ditunjukkan oleh penurunan nilai loss—baik *bounding box loss, classification loss*, maupun *objectness loss*. Hal ini menandakan bahwa model semakin mampu memahami pola visual objek api di dalam dataset.

4.2 Evaluasi Model

Evaluasi dilakukan untuk mengetahui seberapa baik model mendeteksi objek api dalam data uji. Evaluasi dilakukan menggunakan metrik YOLO standar, yaitu mAP50 dan mAP50-95. Model YOLOv8 menghasilkan performa seperti Tabel 2:

Tabel 2. Hasil Evaluasi

Metrik	Nilai
mAP50	0.646
mAP50-95	0.391
Precision	0.889
Recall	0.455
Inference Speed	282.5 *ms/frame*

Nilai recall yang diperoleh pada penelitian ini sebesar 0.455 menunjukkan bahwa masih terdapat objek api yang tidak berhasil terdeteksi oleh model. Nilai recall yang relatif rendah ini mengindikasikan kecenderungan model untuk lebih selektif dalam mendeteksi objek api. Kondisi tersebut dapat disebabkan oleh keterbatasan jumlah dataset yang digunakan, yaitu hanya 100 citra, serta tingginya variasi visual api, seperti ukuran api yang kecil, intensitas cahaya rendah, atau sebagian objek api tertutup oleh latar belakang. Meskipun demikian, nilai precision yang tinggi (0.889) menunjukkan bahwa sebagian besar deteksi yang dihasilkan model merupakan deteksi yang benar.

Dalam pengujian model, potensi kesalahan deteksi juga diamati dalam bentuk false negative dan false positive. False negative terjadi ketika objek api berukuran kecil atau berada pada kondisi pencahayaan ekstrem tidak terdeteksi oleh model. Sementara itu, false positive dapat muncul ketika objek non-api memiliki karakteristik visual yang menyerupai api, seperti pantulan cahaya, cahaya lampu berwarna jingga, atau objek dengan tekstur menyerupai nyala api. Temuan ini menunjukkan bahwa meskipun model memiliki tingkat presisi tinggi, masih diperlukan peningkatan kemampuan generalisasi untuk mengurangi kesalahan deteksi.

4.3 Perbandingan dengan Penelitian Sebelumnya

Hasil penelitian ini menunjukkan bahwa nilai precision yang diperoleh lebih tinggi dibandingkan beberapa penelitian sebelumnya yang menggunakan model YOLOv5 dan YOLOv8 untuk deteksi api pada skenario serupa [8], [10]. Namun, nilai recall yang dihasilkan masih lebih rendah, yang menunjukkan bahwa skala dan keberagaman dataset sangat berpengaruh terhadap kemampuan model dalam mendeteksi seluruh objek api. Dibandingkan studi sebelumnya yang menggunakan dataset berskala besar, penelitian ini menegaskan bahwa keterbatasan data dapat berdampak pada sensitivitas deteksi meskipun model yang digunakan tergolong modern.

4.4 Implementasi Sistem Deteksi *Real-Time*

Sistem *real-time* dibangun menggunakan Python, OpenCV, dan model YOLOv8 yang telah dilatih. Kamera webcam menjadi sumber input citra, yang kemudian diproses *frame-by-frame*. Setiap frame dikirimkan ke model untuk dideteksi apakah terdapat objek api.

Gambar 3. Hasil prediksi model terhadap citra asli *dataset* (benar terdeteksi sebagai api)

Gambar 3 menunjukkan hasil pengujian model pada citra uji. Sistem berhasil mendeteksi keberadaan api dan memberikan *bounding box* beserta tingkat *confidence* sebesar 0.50. Hal ini menunjukkan bahwa model mampu mengenali pola visual api pada kondisi lingkungan nyata.

5. Kesimpulan

Penelitian ini menunjukkan bahwa model YOLOv8 mampu digunakan untuk mendeteksi api secara real-time berbasis pengolahan citra dengan performa yang cukup baik. Hasil evaluasi menunjukkan nilai mAP50 sebesar 0.646, precision 0.889, dan recall 0.455, yang mengindikasikan bahwa model memiliki tingkat akurasi deteksi yang tinggi meskipun sensitivitas deteksi masih perlu ditingkatkan. Implementasi sistem menggunakan OpenCV dan webcam juga membuktikan bahwa YOLOv8 dapat diterapkan secara responsif pada sistem pemantauan kebakaran berbasis kamera.

Namun demikian, penelitian ini memiliki beberapa keterbatasan. Jumlah dataset yang digunakan relatif terbatas, yaitu hanya 100 citra api, sehingga berpengaruh terhadap kemampuan generalisasi model dan nilai recall yang dihasilkan. Selain itu, variasi kondisi lingkungan pada dataset masih belum sepenuhnya merepresentasikan seluruh skenario kebakaran di dunia nyata, seperti kondisi asap tebal, pencahayaan ekstrem, atau sudut pandang kamera yang beragam.

Sebagai agenda pengembangan lanjutan, penelitian selanjutnya disarankan untuk menggunakan dataset yang lebih besar dan beragam, melakukan perbandingan kinerja dengan model deteksi objek lain seperti YOLOv5 atau EfficientDet, serta mengoptimalkan teknik data augmentation dan penyesuaian parameter model. Selain itu, integrasi sistem dengan perangkat keras tambahan atau sistem peringatan dini juga dapat dipertimbangkan untuk meningkatkan efektivitas penerapan pada lingkungan nyata.

Ucapan Terima kasih

Penulis mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan dan dukungan selama proses penulisan artikel ini. Ucapan terima kasih khusus disampaikan kepada pihak-pihak yang telah membantu baik dari segi teknis maupun dalam hal pembiayaan. Segala bentuk kontribusi yang diberikan sangat berarti bagi tersusunnya artikel ini dengan baik.

Daftar Pustaka

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, June 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2015, *arXiv*. doi: 10.48550/ARXIV.1512.03385.

[3] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020, *arXiv*. doi: 10.48550/ARXIV.2004.10934.

[4] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," 2015, doi: 10.48550/ARXIV.1512.02325.

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

[6] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," 2017, *arXiv*. doi: 10.48550/ARXIV.1708.02002.

[7] C. Jia, D. Wang, J. Liu, and W. Deng, "Performance Optimization and Application Research of YOLOv8 Model in Object Detection," *Acad. J. Sci. Technol.*, vol. 10, no. 1, pp. 325–329, Mar. 2024, doi: 10.54097/p9w3ax47.

[8] S. Duan, J. Zhou, and C. Liu, "A lightweight YOLOv8 algorithm for real-time flame detection in fire," Aug. 26, 2024, *In Review*. doi: 10.21203/rs.3.rs-4823368/v1.

[9] L. Lei, R. Duan, F. Yang, and L. Xu, "Low Complexity Forest Fire Detection Based on Improved YOLOv8 Network," *Forests*, vol. 15, no. 9, p. 1652, Sept. 2024, doi: 10.3390/f15091652.

[10] N. D. Ismail, R. Ramli, and M. N. Ab Rahman, "Evaluating YOLOv5s and YOLOv8s for Kitchen Fire Detection: A Comparative Analysis," *Emit. Int. J. Eng. Technol.*, vol. 12, no. 2, pp. 167–181, Dec. 2024, doi: 10.24003/emitter.v12i2.882.