Teknika 18 (2): 455 – 465

JURNAL TEKNIKA
ISSN: 0854-3143
e-ISSN: 2622-3481
Journal homepage: http://jurnal.polsri.ac.id/index.php/teknika
Journal Email: teknika@polsri.ac.id

AIoT untuk Deteksi Dini Kebakaran Menggunakan Integrasi Komputasi Tepi dan Kamera IP

Dona Yuliawati [*1], Sushanty Saleh[2], TM Zaini[3]

[*1,2,3]Jurusan Sistem Informasi, Institut Informatika dan Bisnis Darmajaya, Bandar Lampung, Indonesia
*Email Penulis Korespondensi: **[*1]**donayuliawati@darmajaya.ac.id**

Abstrak

Berdasarkan data Badan Nasional Penanggulangan Bencana bencana kebakaran di Indonesia terjadi sebanyak 4.727 kali antara tahun 2015-2024. Oleh karena itu diperlukan teknologi untuk mendeteksi sumber api secara cepat dan akurat. Oleh karena itu pada penelitian ini kami mengembangkan sistem deteksi sumber kebakaran dengan menggunakan teknologi YOLOv8 pada platform komputer tepi Jetson untuk mendeteksi kebakaran secara real-time dan akurat. Hasil penelitian menunjukkan bahwa model kami berhasil mendeteksi api menggunakan Jetson dan aliran streaming dari Kamera IP dengan evaluasi confusion matrix menunjukkan akurasi deteksi sebesar 96%, Precision-Recall dengan presisi rata-rata 93,5% dan recall rata-rata 97%, serta mean Average Precision (mAP) sebesar 97,90%. Grafik F1-Score menunjukkan akurasi berada di 97%. Hasil ini menunjukkan bahwa model efektif untuk deteksi kebakaran real-time, dengan keseimbangan baik antara presisi dan recall, menjadikannya solusi potensial untuk diimplementasikan pada pendeteksian api secara real-time dengan AIoT.

Kata kunci*—YOLO, Deteksi API, AIOT, Jetson, Kamera IP*

Abstract

According to data from the National Disaster Management Agency, fire disasters in Indonesia occurred 4,727 times between 2015-2024. Therefore, technology is needed to detect fire sources quickly and accurately. In this study, we developed a fire source detection system using YOLOv8 technology on the Jetson edge computing platform to detect fires in real-time and accurately. The results of the study show that our model successfully detected fire using Jetson and streaming from IP Cameras, with a confusion matrix evaluation indicating a detection accuracy of 96%, Precision-Recall with an average precision of 93.5% and an average recall of 97%, and a mean Average Precision (mAP) of 97.90%. The F1-Score graph indicates an accuracy of 97%. These results show that the model is effective for real-time fire detection, with a good balance between precision and recall, making it a potential solution for real-time fire detection with AIoT.

Keywords— *YOLO, Fire Detection, AIOT, Jetson, IP Camera*

1. PENDAHULUAN

Kebakaran api adalah bencana yang dapat menyebabkan kerusakan dan kerugian materil yang besar dalam waktu yang singkat. Seperti yang terjadi di Indonesia, berdasarkan data dari Badan Nasional Penanggulangan Bencana, telah terjadi 4.727 bencana kebakaran dalam 2015 – 2024 tahun terakhir [1]. Fenomena ini menggarisbawahi urgensi untuk pengembangan sistem deteksi kebakaran yang lebih responsif, yang dapat membantu mengurangi dampak negatif kebakaran dan mempercepat tanggapan dalam situasi darurat.

Selama ini, telah banyak teknologi, metode, dan satuan operasional yang disusun untuk melakukan pencegahan kebakaran sedini mungkin sehingga kerugian dapat diminimalkan. Namun, kendala-kendala seperti keterbatasan waktu, ketersediaan sumber daya, dan kompleksitas lingkungan sering kali menjadi tantangan dalam upaya pencegahan dan deteksi kebakaran yang efektif. Oleh karena itu, pengembangan sistem deteksi kebakaran yang dapat bekerja secara otomatis dan cepat menjadi sangat penting.

Dalam 3 tahun terakhir, teknologi kecerdasan buatan (AI) telah membuka peluang baru dalam deteksi kebakaran dengan tingkat akurasi dan kecepatan yang lebih tinggi, seperti pada tahun 2022, Seydi *et al.* mengembangkan *deep learning framework* yang diberi nama *fire-net* untuk mendeteksi kebakaran dini di hutan [2]. Kemudian pada tahun yang sama Avazov *et al.* mengembangkan deteksi kebakaran dengan YOLOv4 yang dapat mendeteksi kebakaran secara cepat baik dalam keadaan malam, siang hari atau dalam keadaan hujan [3]. Kemudian pada tahun 2023, Wu S *et al.* mengembangkan deteksi kebakaran dini dengan *transformers* dengan hasil deteksi adalah 94% [4]. Aral *et al.* juga mengembangkan deteksi pada kebakaran hutan dengan *UAV Drone* dan dengan *deep learning* akurasi sebesar 92% [5]. Pada tahun 2024, Gao *et al.* melakukan peningkatan akurasi dan presisi pada YOLOv8 untuk mendeteksi asap dan api yang diberi nama YOLOv8n-EBF dengan mendapatkan presisi deteksi sebesar 77,8% lebih tinggi 7% dari YOLOv8 sebelumnya [6]. Zhang pada tahun 2024 juga mengembangkan konsep model deteksi asap dan api pada IOT *surviliennce camera* dan mendapatkan akurasi deteksi sebesar 86% [7].

Dari semua penelitian tersebut, untuk saat ini, penelitian yang berfokus pada pengembangan deteksi kebakaran di platform *edge computing* Jetson masih jarang dilakukan, oleh karena itu menyisakan *gap* penelitian yang dapat diisi. Penggunaan Jetson sebagai *platform edge computing* menawarkan potensi yang signifikan untuk implementasi deteksi kebakaran secara *real-time* dengan efisiensi daya yang lebih tinggi, sehingga penelitian ini bertujuan untuk mengisi kekosongan tersebut dan mengembangkan solusi yang lebih efektif dalam konteks *AIoT*. Lebih lanjut, Implementasi YOLOv8 dengan komputer tepi dan IP *camera* dapat menjadi pendekatan yang baru dalam deteksi api, sehingga akan memberikan kontribusi bagaimana pemanfaatan komputer tepi, IP camera dapat meningkatkan kecepatan dalam deteksi kebakaran api.

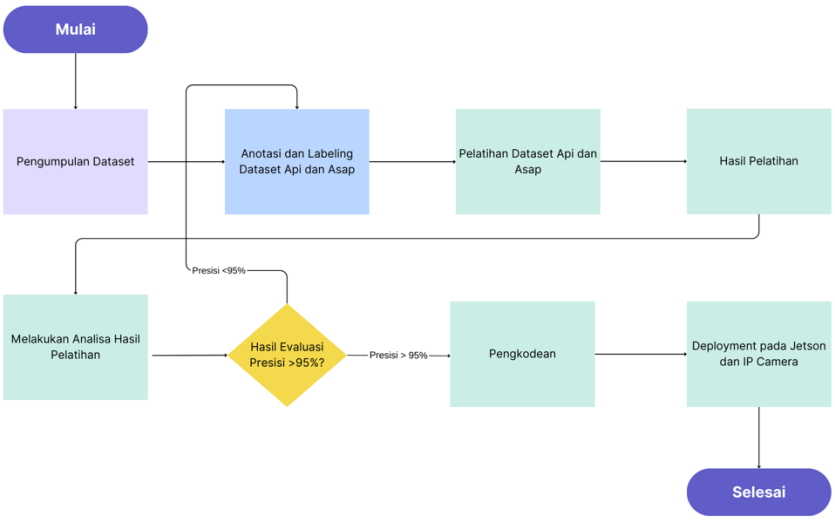
2. METODE PENELITIAN

Pada bagian ini kami akan menjelaskan metodologi penelitian dan arsistektur yang digunakan untuk mengembangkan model deteksi api YOLOv8 dengan menggunakan komputer tepi. YOLO merupakan algoritma deteksi objek yang dikembangkan semenjak tahun 2015 oleh redmon J [8]. Pada masa pengembangan YOLO digunakan untuk mendeteksi objek-objek seperti anjing, kucing, kursi, mobil, sepeda, dll. Selanjutnya sampai saat ini YOLO terus dikembangkan oleh *ultralytics* [9], dan memiliki performa secara signifikan di setiap variannya [10], [11].

2.1 Metode Penelitian

Penelitian ini bertujuan untuk mengembangkan dan mengimplementasikan model kecerdasan buatan (AI) dalam mendeteksi api dan asap menggunakan *dataset* yang relevan. Proses ini mencakup berbagai tahap mulai dari pengumpulan data hingga penerapan model AI pada

perangkat keras di lapangan. Pada Gambar 1, merupakan penjelasan dari metode penelitian yang diterapkan.



Gambar 1 Metode Penelitian

1. Langkah pertama dalam penelitian ini adalah mengumpulkan *dataset* yang berisi gambar dan video yang menunjukkan api dan asap. Dataset ini akan digunakan sebagai bahan dasar untuk melatih model AI.
2. *Dataset* yang telah dikumpulkan kemudian diberi anotasi dan label secara manual untuk menandai area yang mengandung api dan asap. Proses ini penting untuk memastikan bahwa model AI dapat belajar dengan akurasi tinggi dari data yang sudah terlabel.
3. *Dataset* yang sudah diberi anotasi dan label kemudian digunakan untuk melatih model AI.
4. Setelah proses pelatihan selesai, model AI akan menghasilkan hasil pelatihan yang mencakup kemampuan model dalam mendeteksi api dan asap dari dataset yang diberikan.
5. Hasil pelatihan kemudian dievaluasi untuk memastikan presisi model dalam mendeteksi api dan asap. Jika presisi yang dihasilkan lebih dari 95%, maka proses akan dilanjutkan ke tahap berikutnya. Jika tidak, model akan kembali ke tahap pengumpulan *dataset* untuk peningkatan lebih lanjut.
6. Dalam kasus dimana presisi kurang dari 95%, dilakukan analisa hasil pelatihan untuk mengidentifikasi kelemahan model dan menentukan langkah-langkah perbaikan yang diperlukan.
7. Setelah model mencapai presisi yang diinginkan, model AI yang telah dilatih dikodekan dan diintegrasikan dengan sistem yang akan digunakan.
8. Model AI yang telah dikodekan kemudian diterapkan pada perangkat keras seperti Jetson Nano dan IP *Camera*. Proses ini memungkinkan implementasi model AI dalam lingkungan nyata untuk melakukan deteksi api dan asap secara real-time.

2.2 Dataset

Dataset adalah kumpulan data yang terorganisir dan terstruktur yang digunakan untuk tujuan analisis atau penelitian [12]. Dalam konteks YOLO, *dataset* merujuk pada kumpulan gambar yang digunakan untuk melatih dan menguji model deteksi objek berbasis YOLO [13]. Untuk mendapatkan model yang baik, kami menggunakan *dataset* api yang telah dianotasi di *roboflow* [14]. Tujuan anotasi sendiri adalah untuk memberikan label objek api yang akan dikenali oleh YOLO. Dalam konteks penelitian ini, kami menggunakan 782 gambar *dataset*,

dengan 1 *class* yang Bernama *fire* atau api. Untuk memperkuat sensitifitas AI dalam mendeteksi api, maka setiap gambar kami augmentasi 3x [15] dengan *flip vertical*, sehingga total gambar menjadi 1232 gambar pada *dataset*. Gambar-gambar pada *dataset* tersebut selanjutnya dibagi menjadi 3 bagian yaitu, 86% atau sekitar 1061 gambar untuk pelatihan, 8% atau 103 gambar untuk validasi hasil pelatihan, dan 6% atau 68 gambar untuk uji coba saat pelatihan.

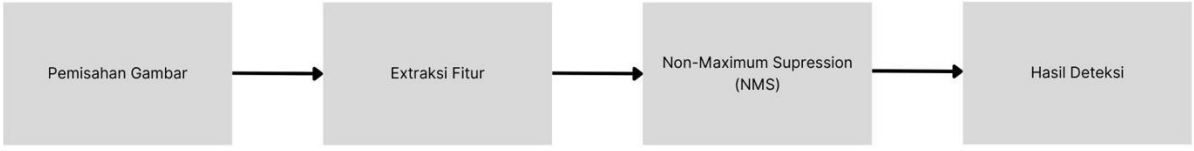
2.3 Parameter Pelatihan

Sesuai dengan dokumentasi yang diberikan oleh *ultralytics* sebagai *framework* induk YOLOv8, bahwa akurasi model semakin baik ketika jumlah *epoch* pelatihan dan *batch* semakin tinggi [16], maka pada pelatihan *dataset* ini kami menggunakan parameter 200 *epoch* pelatihan dan 12 *batch*. Dalam konteks pembelajaran *neural network*, *epoch* merujuk pada satu siklus penuh yang digunakan untuk melatih model [17]. Dalam setiap *epoch*, model akan melihat setiap contoh pada data pelatihan dan melakukan proses pembelajaran berdasarkan data tersebut [18], sehingga jika ada 1061 *dataset* maka dalam 1x *epoch*, model akan melihat 1061x contoh objek api pada *dataset*, dengan demikian dikarenakan kami menggunakan 200 *epoch*, maka model akan terlatih mengenali objek api sebanyak 212.200x. Kemudian *batch* merujuk kepada pembagian data pelatihan setiap iterasi pembelajaran. Semakin tinggi jumlah *batch* maka semakin banyak contoh data yang diproses secara bersamaan [19], sehingga memungkinkan model untuk belajar dari variasi yang lebih besar dari data pelatihan dalam setiap iterasi [20]. Hal ini dapat membantu model untuk mengenali pola-pola yang lebih kompleks dan dapat menghasilkan pembelajaran yang lebih baik [21]. Namun penggunaan *batch* yang tinggi akan berpengaruh terhadap alokasi penggunaan memori pada GPU, yang dapat menyebabkan *insufficient memory* saat pelatihan. Oleh karena itu, kami menggunakan 12 *batch* dengan ukuran gambar 640x640 px, dengan rata-rata ukuran gambar adalah 300kb, dengan asumsi alokasi memori yang dihabiskan adalah 6 Gb / epoch pelatihan.

2.4 Metode Pada YOLOv8

YOLOv8 adalah iterasi teraru dari varian keluarga algoritma you look only once (YOLO) [22], yang dikenal karena kecepatan dan akurasi algoritmanya. Metode YOLO beroperasi dengan mengintegrasikan deteksi objek dan perhitungan bounding box dalam satu langkah. Berikut adalah langkah-langkah umum dari cara kerja YOLO.

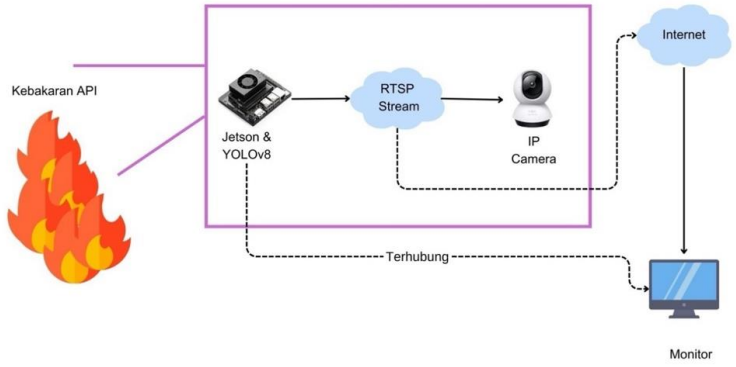
1. Pada langkah pertama, akan dilakukan pemisahan gambar yang akan dideteksi oleh algoritma YOLO.
2. Selanjutnya gambar akan diolah melalui serangkaian lapisan konvolusi untuk mengekstrak fitur-fitur penting dari gambar
3. Kemudian gambar dibagi menjadi *grid cell*, dimana setiap *cell* akan memiliki *confidence score* yang menunjukkan seberapa yakin model bahwa kotak tersebut berisi objek yang relevan.
4. Terakhir, YOLO menggunakan NMS (*non maximum surpression*) untuk menghapus kotak pembatas dengan kepercayaan rendah dan hanya menyimpan kotak pembatas dengan kepercayaan tinggi.



Gambar 2 Proses Cara Kerja YOLO

2.5 Arsitektur Sistem

Arsitektur Sistem Penelitian ini bertujuan mengembangkan model deteksi api dengan mengimplementasikan algoritma YOLO di *Edge AI Computing* dalam lingkungan *AIoT* (*Artificial Intelligence of Things*) untuk mendeteksi insiden kebakaran api sedini mungkin.



Gambar 3 Arsitektur Sistem

Pada Gambar 3, arsitektur sistem kami menggunakan Jetson Nano Orin yang telah terintegrasi dengan YOLOv8. Kemudian YOLO akan mengirimkan *feed* IP Camera melalui protocol RTSP (*Real Time Streaming protocol*) ke monitor [23], sehingga, model dapat mendeteksi api melalui IP *Camera* yang terpasang secara *realtime*.

2.6 Metode Evaluasi

Dalam penelitian ini, evaluasi performa model AI untuk mendeteksi api dan asap dilakukan menggunakan beberapa metrik evaluasi yaitu *precision, recall, accuracy, dan F1-measure*. Metrik-metrik ini dihitung berdasarkan *confusion matrix*, yang merupakan matriks yang menunjukkan jumlah hasil prediksi benar dan salah dari model.

1. *Confusion matrix* adalah tabel yang digunakan untuk mengukur kinerja model klasifikasi. Matriks ini terdiri dari empat komponen b:
   - *True Positive* (TP): Jumlah data positif yang diprediksi benar oleh model.
   - *True Negative* (TN): Jumlah data negatif yang diprediksi benar oleh model.
   - *False Positive* (FP): Jumlah data negatif yang diprediksi salah sebagai positif oleh model.
   - *False Negative* (FN): Jumlah data positif yang diprediksi salah sebagai negatif oleh model.
2. *Precision* akan menghitung rasio jumlah *true positive* dengan jumlah total prediksi positif (*true positive* dan *false positive*). *Precision* menunjukkan seberapa akurat model dalam memprediksi kelas positif [24].

   $$precison = \frac{tp}{tp+fp} \quad (1)$$
3. *Recall* akan menghitung rasio antara jumlah *true positive* dengan jumlah total data sebenarnya positif (*true positive* dan *false negative*). *Recall* menunjukkan kemampuan model untuk menemukan semua data positif, sehingga dapat digunakan untuk mengevaluasi kemampuan model dalam mendeteksi semua data positif [25].

   $$precison = \frac{tp}{tp+fp} \quad (2)$$
4. *F1-Measure* dan *precision-recall* adalah rata-rata harmonis dari *precision* dan *recall*. Metrik ini memberikan gambaran seimbang antara *precision* dan *recall*, terutama jika ada ketidakseimbangan antara jumlah data positif dan negatif [26].

   $$f1 - measure = \frac{2.Precision.Recall}{precision+recall} \quad (3)$$

3. HASIL DAN PEMBAHASAN

3.1 Hasil Dan Pembahasan

Model yang telah kami latih dihasilkan dari 782 Gambar Api yang bersumber pada *roboflow* [14]. Pada tahap awal experimen, kami menggunakan *google colab* untuk melakukan pelatihan *dataset* dengan 200 *epoch* dan 12 *batch*. Selanjutnya kami melakukan uji coba di komputasi tepi NVIDIA Jetson Nano dengan 8Gb Ram, 1024-Core NVIDIA Ampere Architechture GPU dengan 32 Tensor Cores [27] dan Ip *Camera* TAPO C200.

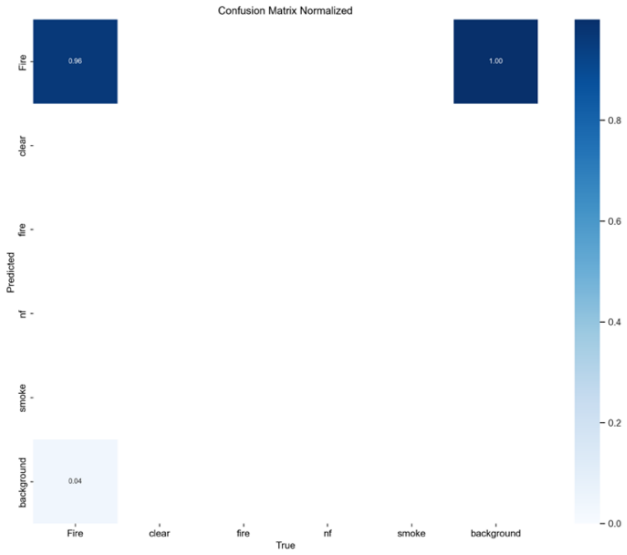
Gambar 4 Experimen Sistem

Untuk melakukan uji coba, kami menggunakan IP pada protokol *realtime streaming protocol* (RTSP) yang disisipkan pada kode aplikasi sebagai pengganti kamera. Selanjutnya, YOLO akan memproses dan menganalisasi tangkapan video dari IP Kamera untuk dideteksi pusat api. Pada Gambar 5, sistem mampu mendeteksi keberadaan api dengan kestabilan 12 FPS. Selain itu YOLO dapat mendeteksi keberadaan api dengan tingkat keyakinan diatas 0,5 (50%).

Gambar 5 Deteksi API dengan Jetson, YOLO dan Kamera IP

3.2 Evaluasi pada Confusion Matrix

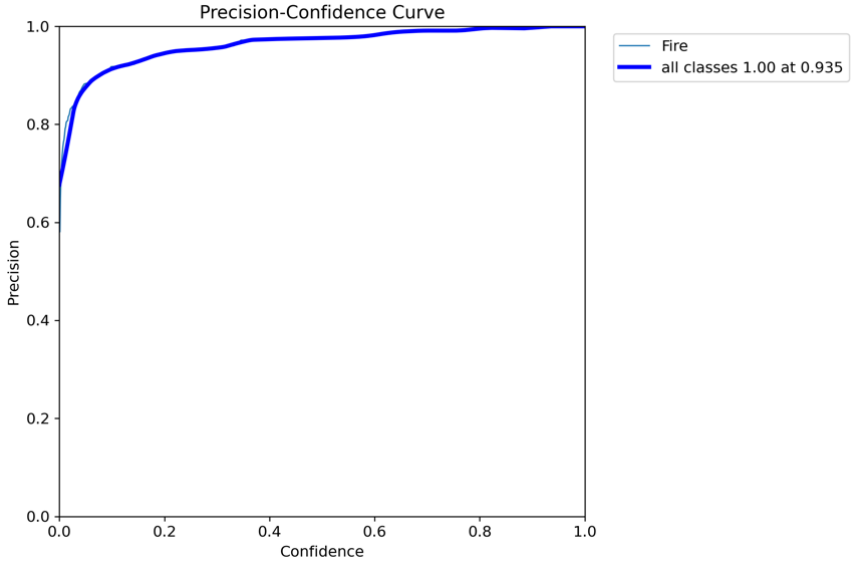
Grafik Confusion Matrix yang dinormalisasi pada Gambar 6 menunjukkan bahwa model memiliki akurasi tinggi dalam mendeteksi kelas "*Fire*" dan "*clear*" dengan nilai akurasi masing-masing sebesar 96% dan 100%. Namun, terdapat beberapa kesalahan klasifikasi pada kelas "*background*", di mana sekitar 4% dari prediksi "*background*" salah terdeteksi sebagai "*Fire*". Tidak ada prediksi yang salah untuk kelas "*clear*" dan "*Fire*" yang berlabel lain.



Gambar 6 *Confusion Matrix*

3.3 Precision-Confidence Curve

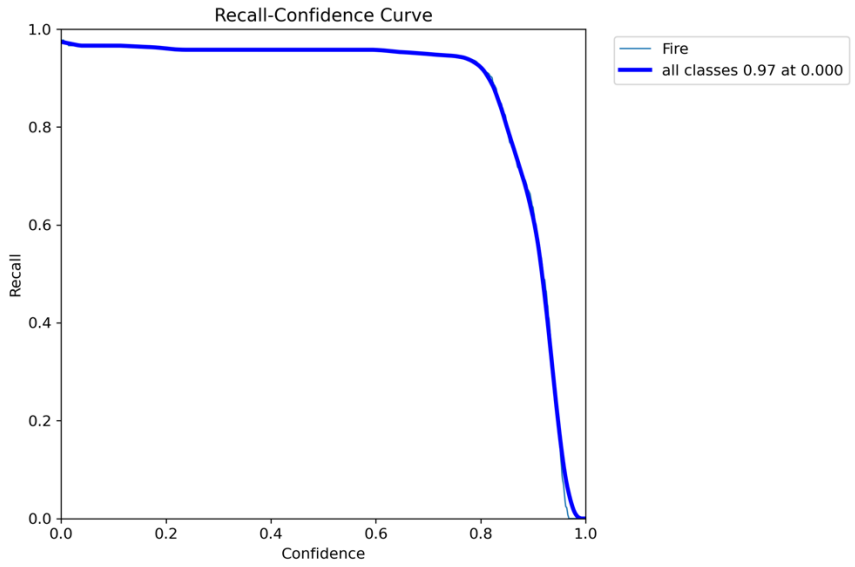
Grafik *Precision-Confidence Curve* pada grafik Gambar 7 dengan menggunakan rumus (1) menunjukkan bahwa model kami memiliki presisi yang sangat tinggi pada *confidence score* yang tinggi. Presisi rata-rata keseluruhan mencapai 93,5%, dengan nilai presisi mendekati 100% pada *confidence score* mendekati 1. Grafik ini menunjukkan peningkatan presisi seiring dengan meningkatnya *confidence score*.



Gambar 7 Grafik *Precision-Confidence Curve*

3.3 Recall-Confidence Curve

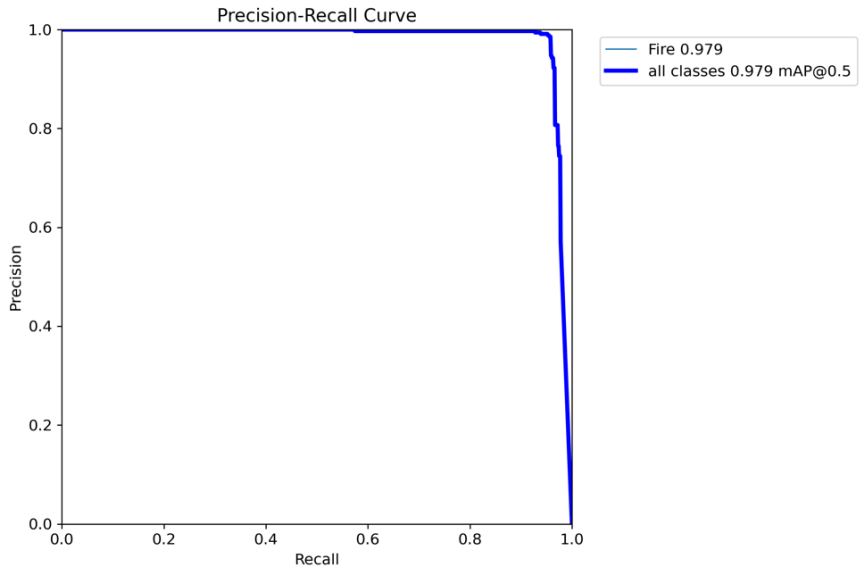
Grafik *Recall-Confidence Curve* pada Gambar 8 yang menggunakan dengan menggunakan rumus (2) menunjukkan bahwa model memiliki *recall* yang tinggi pada *confidence score* yang rendah hingga sedang, dengan *recall* rata-rata 97%. Pada *confidence score* mendekati 1, *recall* menurun drastis karena model menjadi lebih ketat dalm membuat prediksi saat sangat yakin, sehingga mengurangi jumlah prediksi benar (*True Positives*).



Gambar 8 Grafik *Precision-Confidence Curve*

3.4 Precision-Recall Curve

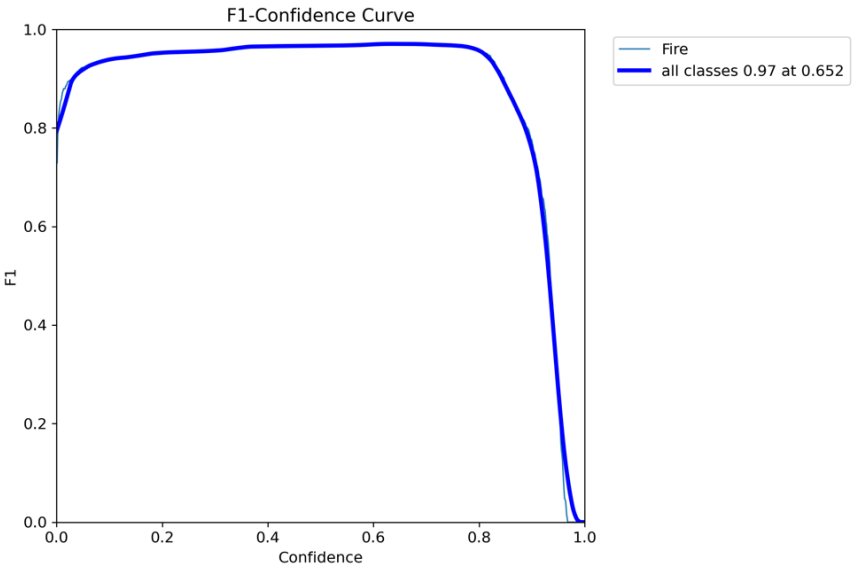
Grafik *Precision-Recall Curve* pada Gambar 9 dengan menggunakan rumus (3) menunjukkan bahwa model memiliki performa yang sangat baik dengan *precision dan recall* yang tinggi dengan nilai *mean Average Precision* (mAP) sebesar 0,979 atau 97,90% pada *threshold* 0,5. Ini berarti bahwa model mampu mendeteksi api dengan sangat akurat tanpa banyak kesalahan prediksi positif.



Gambar 9 Grafik *Precision-Recall Curve*

3.5 F1-Score-Confidence Curve

Grafik F1-*Confidence Curve* pada Gambar 10 yang menggunakan rumus (3) menunjukkan bahwa model kami memiliki kinerja yang sangat baik dengan nilai F1 adalah 0,97 pada confidence score 0,652. Ini mengindikasikan bahwa model kami mampu menjaga keseimbangan antara presisi dan *recall* dengan sangat baik pada *confidence score* rendah hingga sedang.



Gambar 10 F1-*Confidence Curve*

Penurunan nilai F1 pada *confidence score* tinggi menunjukkan bahwa model menjadi lebih selektif dalam membuat prediksi, yang dapat mengurangi jumlah prediksi positif yang benar (*True Positives*) untuk memastikan bahwa prediksi yang dibuat sangat akurat. Hasil ini menegaskan bahwa model kami sangat baik dalam menghasilkan prediksi yang seimbang antara *presisi* dan *recall* pada *confidence score* yang seimbang.

4. KESIMPULAN

Hasil penelitian ini menyoroti pentingnya deteksi kebakaran dini dan efektivitas teknologi AI dalam mencapai tujuan tersebut. Berdasarkan analisis dan eksperimen yang dilakukan, beberapa kesimpulan utama dapat diambil sebagai berikut:

1. Penggunaan YOLOv8 pada platform *edge computing* Jetson memungkinkan deteksi kebakaran secara *real-time* dengan Kamera IP dengan akurasi deteksi mencapai 97 – 98%.
2. Secara keseluruhan FPS deteksi dengan Kamera IP hanya mencapai 10 – 12 FPS.
3. Model yang kami kembangkan dapat bersaing dengan algoritma lain untuk deteksi api, seperti transformers dengan akurasi 94% [5] , *Deep Learning* dengan akurasi 86 - 92% [5] [7], serta YOLOv8n-EBF yang memiliki akurasi dengan 78% [6].
4. Model kami belum diuji coba pada ketinggian Kamera IP 2-3 meter, sehingga belum diketahui efektifitas model ini pada saat melakukan deteksi objek kecil.

5. SARAN

YOLOv8 dengan *Jetson* Nano Orin memiliki potensi untuk diterapkan untuk memberikan notifikasi deteksi api secara dini, sehingga kebakaran bisa ditanggulangi secepatnya. Namun, FPS yang rendah ketika dijalankan di *Jetson* menjadi *gap* penelitian yang harus ditingkatkan, oleh karena itu, pada penelitian selanjutnya kami akan melakukan pemangkasan *neural network* pada YOLOv8, sehingga akan menjadi lebih ringan, dan dapat meningkatkan FPS pada deteksi melalui Kamera IP. Selain itu, melakukan *tuning* pada *small object* bisa menjadi improvisasi selanjutnya, sehingga YOLO masih dapat mendeteksi api saat Kamera IP berada di ketinggian 2-3 meter.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tim Redaksi Jurnal Teknika Politeknik Negeri Sriwijaya yang telah memberi memberi kesempatan, sehingga artikel ilmiah ini dapat diterbitkan. Selain itu kami mengucapkan pada IIB Darmajaya ada bantuan penelitian yang diberikan.

DAFTAR PUSTAKA

[1] Badan Nasional Penanggulangan Bencana (BNPB), "Data Informasi Bencana Indonesia (DIBI)." Accessed: May 04, 2024. [Online]. Available: https://dibi.bnpb.go.id/home/index2

[2] S. T. Seydi, V. Saeidi, B. Kalantar, N. Ueda, and A. A. Halin, "Fire-Net: A Deep Learning Framework for Active Forest Fire Detection," *J Sens*, vol. 2022, p. 8044390, Jan. 2022, doi: 10.1155/2022/8044390.

[3] K. Avazov, M. Mukhiddinov, F. Makhmudov, and Y. I. Cho, "Fire Detection Method in Smart City Environments Using a Deep-Learning-Based Approach," *Electronics 2022, Vol. 11, Page 73*, vol. 11, no. 1, p. 73, Dec. 2021, doi: 10.3390/ELECTRONICS11010073.

[4] S. Wu, B. Sheng, G. Fu, D. Zhang, and Y. Jian, "Multiscale fire image detection method based on CNN and Transformer," *Multimed Tools Appl*, pp. 1–25, Oct. 2023, doi: 10.1007/S11042-023-17482-4/TABLES/13.

[5] R. A. Aral, C. Zalluhoglu, and E. Akcapinar Sezer, "Lightweight and attention-based CNN architecture for wildfire detection using UAV vision data," *Int J Remote Sens*, vol. 44, no. 18, pp. 5768–5787, Jan. 2023, doi: 10.1080/01431161.2023.2255349.

[6] P. Gao, "A Fire and Smoke Detection Model Based on YOLOv8 Improvement," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 3, pp. 179–190, Jan. 2024, doi: 10.14569/IJACSA.2024.0150318.

[7] D. Zhang, "A Yolo-based Approach for Fire and Smoke Detection in IoT Surveillance Systems," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 1, pp. 87–94, Jan. 2024, doi: 10.14569/IJACSA.2024.0150109.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 779–788, Jun. 2015, doi: 10.1109/CVPR.2016.91.

[9] C. Y. Yang, Y. N. Lin, V. R. L. Shen, F. H. C. Shen, and C. C. Wang, "A Novel IoT-Enabled System for Real-Time Face Mask Recognition Based on Petri Nets," *IEEE Internet Things J*, vol. 11, no. 4, pp. 6992–7001, Feb. 2024, doi: 10.1109/JIOT.2023.3313583.

[10] N. F. Alhussainan, B. Ben Youssef, and M. M. Ben Ismail, "A Deep Learning Approach for Brain Tumor Firmness Detection Based on Five Different YOLO Versions: YOLOv3–YOLOv7," *Computation*, vol. 12, no. 3, p. 44, Mar. 2024, doi: 10.3390/computation12030044.

[11] M. Hussain, "YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection," *Machines 2023, Vol. 11, Page 677*, vol. 11, no. 7, p. 677, Jun. 2023, doi: 10.3390/MACHINES11070677.

[12] R. Nazeer *et al.*, "Detection of cotton leaf curl disease's susceptibility scale level based on deep learning," *Journal of Cloud Computing*, vol. 13, no. 1, p. 50, Dec. 2024, doi: 10.1186/s13677-023-00582-9.

[13] B. Deng, Y. Lu, and J. Xu, "Weed database development: An updated survey of public weed datasets and cross-season weed detection adaptation," *Ecol Inform*, vol. 81, p. 102546, Jul. 2024, doi: 10.1016/j.ecoinf.2024.102546.

[14] Fire, "Fire Detection Dataset." Feb. 2024. [Online]. Available: https://universe.roboflow.com/fire-r56mr/fire-detection-ma7vm

[15] M. J. Sundari and N. C. Brintha, "TLOD: Innovative ovarian tumor detection for accurate multiclass classification and clinical application," *Network Modeling Analysis in Health Informatics and Bioinformatics*, vol. 13, no. 1, p. 18, Dec. 2024, doi: 10.1007/s13721-024-00454-5.

[16] Ultralytics, "Tips for Best Training Results - Ultralytics YOLOv8 Docs." Accessed: May 02, 2024. [Online]. Available: https://docs.ultralytics.com/yolov5/tutorials/tips_for_best_training_results

[17] T. Özer and Ö. Türkmen, "An approach based on deep learning methods to detect the condition of solar panels in solar power plants," *Computers and Electrical Engineering*, vol. 116, p. 109143, May 2024, doi: 10.1016/j.compeleceng.2024.109143.

[18] E. Casas, L. Ramos, E. Bendek, and F. Rivas-Echeverria, "YOLOv5 vs. YOLOv8: Performance Benchmarking in Wildfire and Smoke Detection Scenarios," *Journal of Image and Graphics*, vol. 12, no. 2, pp. 127–136, Jan. 2024, doi: 10.18178/joig.12.2.127-136.

[19] E. C. Tetila *et al.*, "YOLO performance analysis for real-time detection of soybean pests," *Smart Agricultural Technology*, vol. 7, p. 100405, Mar. 2024, doi: 10.1016/j.atech.2024.100405.

[20] H. R. Suresh, M. Shanmuganathan, T. Senthilkumar, and B. S. Vidhyasagar, "Deep learning-based image forgery detection system," *International Journal of Electronic Security and Digital Forensics*, vol. 16, no. 2, pp. 160–172, Jan. 2024, doi: 10.1504/IJESDF.2024.137036.

[21] T. S. Chi *et al.*, "Enhancing EfficientNet-YOLOv4 for Integrated Circuit Detection on Printed Circuit Board (PCB)," *IEEE Access*, vol. 12, pp. 25066–25078, Jan. 2024, doi: 10.1109/ACCESS.2024.3359639.

[22] M. Sohan, T. S. Ram, C. Venkata, and R. Reddy, "A Review on YOLOv8 and Its Advancements," pp. 529–545, 2024, doi: 10.1007/978-981-99-7962-2_39.

[23] M. Xie, L. Wang, M. Ma, and P. Zhang, "Performance Evaluation Method for Intelligent Computing Components for Space Applications," *Sensors*, vol. 24, no. 1, p. 145, Jan. 2024, doi: 10.3390/s24010145.

[24] M. Hussain, "YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection," *Machines 2023, Vol. 11, Page 677*, vol. 11, no. 7, p. 677, Jun. 2023, doi: 10.3390/MACHINES11070677.

[25] Z. Dou *et al.*, "An Improved YOLOv5s Fire Detection Model," *Fire Technol*, vol. 60, no. 1, pp. 135–166, Jan. 2024, doi: 10.1007/S10694-023-01492-7/FIGURES/21.

[26] U. Sirisha, S. P. Praveen, P. N. Srinivasu, P. Barsocchi, and A. K. Bhoi, "Statistical Analysis of Design Aspects of Various YOLO-Based Deep Learning Models for Object Detection," *International Journal of Computational Intelligence Systems*, vol. 16, no. 1, pp. 1–29, Dec. 2023, doi: 10.1007/S44196-023-00302-W/TABLES/21.

[27] "Jetson Nano Developer Kit | NVIDIA Developer." Accessed: Jan. 11, 2024. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit