

PROTOTYPE DETEKSI API DI BELT CONVEYOR BATU BARA MENGGUNAKAN YOLOV8 DENGAN NOTIFIKASI TELEGRAM

Adhitya Eka^{1*}, Supatman²

^{1,2}Informatika, Universitas Mercu Buana Yogyakarta; Jl. Ring Road Utara, Ngropoh, Condongcatur, Kec. Depok, Kabupaten Sleman, Daerah Istimewa Yogyakarta 55281

Keywords:

Belt Coveyor;
Batu bara;
YOLOv8;
Telegram

Correspondent

Email:

adhityaeka69@gmail.com

Abstrak. Penelitian ini dilakukan guna menguji performa sistem deteksi api menggunakan YOLOv8 sebagai deteksi dini api pada belt conveyor mulai dari bara api, percikan api, dan api itu sendiri. Sistem ini juga dilengkapi dengan notifikasi bot telegram yang menyertakan informasi lokasi serta bukti visual kejadian secara instan, agar nantinya dapat digunakan untuk mempermudah proses penanganan risiko bahaya dan evakuasi. Hasil eksperimennya menunjukkan bahwa sistem mampu membantu petugas untuk mempermudah proses penanganan risiko bahaya dan evakuasi karena performa sistem yang cepat, akurat, dan notifikasi yang real-time.



Copyright © [JITET](#) (Jurnal Informatika dan Teknik Elektro Terapan). This article is an open access article distributed under terms and conditions of the Creative Commons Attribution (CC BY NC)

Abstract. This study was conducted to evaluate the performance of a fire detection system based on YOLOv8 as an early warning mechanism for belt conveyor fires, including embers, sparks, and open flames. The system is also equipped with a Telegram bot notification feature that instantly provides location information and visual evidence of the incident, enabling faster risk mitigation and evacuation processes. The experimental results demonstrate that the system assists personnel in streamlining hazard risk management and evacuation processes, thanks to its fast and accurate performance and real-time notifications.

1. PENDAHULUAN

Belt conveyor merupakan salah satu alat yang digunakan untuk memindahkan barang atau material dalam berbagai industri[1]. Dalam industri penanganan batubara, kebakaran *belt conveyor* dapat menjadi salah satu pemicu risiko keselamatan kerja tertinggi. Kebakaran tersebut disebabkan oleh interaksi fisik berupa gesekan dari material dan mekanis penggerak. Ketika temperatur dari jalur *belt conveyor* meningkat akibat adanya gesekan, hal tersebut langsung dapat memicu debu dari batubara menjadi *self-combustion* (pembakaran diri

sendiri). Apabila *self-combustion* aktif maka dampaknya adalah bara api akan merambat ke sabuk karet dan mengakibatkan kebakaran *belt conveyor*[2].

Ada beberapa insiden kebakaran yang terjadi di lokasi belt conveyor. Salah satunya adalah insiden kebakaran yang terjadi di Pembangkit Listrik Tenaga Uap (PLTU) Punagaya, Jeneponto, Sulawesi Selatan. AKBP Herry Susanto menjelaskan bahwa api diduga berasal dari batu bara yang menyala di *belt conveyor*. Api itu terjadi akibat adanya gesekan batu bara dan tertiuap angin[3]. Insiden serupa

juga pernah terjadi di tambang batu bara Guizhou, Tiongkok Barat Daya. Investigasi awal menunjukkan bahwa insiden tersebut disebabkan oleh pita pengangkut yang terbakar[4].

Sistem deteksi tradisional seperti sensor asap dan sensor panas masih memiliki kelemahan. Sensor asap rentan terhadap deteksi palsu yang disebabkan oleh faktor debu, uap, ataupun aerosol. Sensor panas seringkali mengalami keterlambatan respon. Di sisi lain, sistem deteksi api menggunakan visual CCTV dapat bekerja secara responsif dibandingkan dengan sensor fisik/*smoke detector*. [5]. Dengan adanya keterbatasan tersebut dapat menyebabkan keterlambatan penanganan kebakaran yang terjadi serta dapat menimbulkan kerusakan yang lebih serius sebelum adanya tindakan perbaikan yang dilakukan.

Adanya perkembangan teknologi saat ini memungkinkan penerapan sistem pemeriksaan otomatis yang menggunakan gambar digital guna mendeteksi objek yang muncul pada *belt conveyor* secara tepat dan akurat. Algoritma YOLO merupakan salah satu metode *deep learning* yang paling populer pada pendekatan *object detection* berbasis CNN (*Convolutional Neural Network*) dan telah banyak digunakan pada bidang industri karena mampu mendeteksi objek secara real time dan memproses keseluruhan gambar dalam satu kali pengamatan[6]. CNN juga terbukti ampuh dalam melakukan klasifikasi citra[7].

Oleh karena itu, penelitian ini dilakukan guna menguji performa sistem deteksi api menggunakan YOLOv8 sebagai deteksi dini api mulai dari bara api, percikan api, dan api itu sendiri. Sistem ini juga dilengkapi dengan notifikasi bot telegram yang menyertakan informasi lokasi serta bukti visual kejadian secara instan, agar nantinya dapat digunakan untuk mempermudah proses penanganan risiko bahaya dan evakuasi.

2. TINJAUAN PUSTAKA

2.1 Computer Vision

Computer Vision dapat didefinisikan sebagai suatu proses pembelajaran komputer dalam menganalisis gambar dan video untuk memperoleh hasil yang diinginkan layaknya visualisasi manusia[8]. Dalam tugas *Computer Vision*, deteksi objek bertujuan untuk

mengklasifikasi kategori dan lokasi objek dengan benar[9]. Salah satu kemampuan *computer vision* adalah mampu memberikan waktu respons yang baik dalam hal mitigasi bencana karena analisisnya dilakukan langsung menggunakan citra digital[10].

2.2 CNN (*Convolution Neural Network*)

CNN (*Convolutional Neural Network*) merupakan arsitektur dari jaringan saraf tiruan yang dirancang untuk mengekstraksi gambar dari lapisan konvolusi secara otomatis[11]. Dikarenakan kemampuannya mendeteksi objek yang rumit secara akurat, sudah jadi hal umum CNN menjadi standar utama dalam pengolahan gambar[12].

2.3 Algoritma YOLOv8

YOLOv8 biasa dikenal sebagai arsitektur *one-stage object detector* yang dikembangkan oleh Ultralytics[13]. YOLOv8 memiliki struktur yang efisien dibanding versi sebelumnya, sehingga YOLOv8 dapat menghasilkan FPS (*Frame Per Second*) yang tinggi pada CPU tanpa mengorbankan nilai akurasi[14].

2.4 Telegram untuk Sistem Peringatan Dini (*Early Warning System*)

Telegram merupakan aplikasi pesan instan yang menyediakan API (*Application Programming Interface*) gratis, yang memungkinkan pengguna dapat mengintegrasikannya dengan suatu sistem melalui bot[15]. Sedangkan, sistem peringatan dini dibuat agar informasi penting bisa cepat sampai ke pihak yang berwenang, sehingga dampak bencana bisa dikurangi. Menggabungkan sistem deteksi visual dengan Telegram Bot API memungkinkan untuk meningkatkan efisiensi dan responsivitas dalam mengindikasikan kebakaran secara *real-time*[16].

3. METODOLOGI PENELITIAN

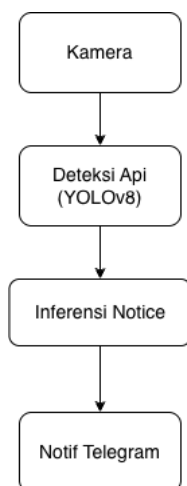
3.1. Jenis dan Pendekatan Penelitian

Jenis penelitian yang digunakan adalah penelitian kuantitatif dengan menggunakan metode eksperimental yang bertujuan untuk menguji kemampuan/performa sistem dalam mendeteksi objek api secara akurat dan cepat dalam mengirim informasi. Penelitian ini dilakukan dengan mengamati respons sistem

dalam mendeteksi objek api dan pengiriman notifikasi.

3.2. Blok Diagram Sistem

Pada penelitian ini perlu melalui beberapa tahapan sistematis, agar deteksi api berjalan optimal di perangkat yang komputasinya terbatas. Adapun alur kerja sistem dari *prototype* deteksi api ini ditunjukkan pada gambar 1.



Gambar 1 Blok diagram deteksi api batubara

Dengan acuan gambar 1, alur dimulai dari akuisisi data yang diperoleh dari kamera. Data tersebut diproses secara *local processing* menggunakan CPU yang terhubung dengan internet. Selanjutnya proses deteksi keberadaan api menggunakan YOLOv8 pada setiap frame. Setelah objek api terdeteksi, secara otomatis sistem akan mengirimkan notifikasi Telegram kepada pengguna yaitu petugas kebakaran atau pihak yang sudah terintegrasi dengan bot Telegram, sehingga tindakan penanganan dapat segera dilakukan.

3.3. Bahan dan Perangkat Penelitian

Pada penelitian eksperimen ini terdapat beberapa bahan dan perangkat yang digunakan untuk menunjang keberlangsungan penelitian. Berikut merupakan kombinasi bahan, perangkat keras dan perangkat lunak yang dipisahkan berdasarkan lingkungannya:

a. Lingkungan Pelatihan (Cloud Computing)

- **Dataset:** Dataset yang digunakan dalam penelitian ini merupakan dataset dari Roboflow *workspace*

sariawan27 *project prototype-coal-flame-detection*.

- **Platform:** Google Collabs (*Free Version*)
- **Akselerator:** GPU NVIDIA Tesla T4 untuk mempermudah dan mempercepat *training model* YOLOv8.
- **Penyimpanan:** Google drive untuk integrasi dataset dari roboflow.

b. Lingkungan Implementasi (Local Computing)

- **Perangkat Keras:** Laptop Macbook Pro 2017, Prosesor intel i7, RAM 16 GB, Intel HD Graphics 630.
- **Perangkat Lunak:** MacOS Ventura, Python 3.11, *Library* Ultralytics (YOLOv8), OpenCV, dan Telegram bot API.

3.4. Prosedur Penelitian

Penelitian ini dibagi tiga fase utama:

a. Fase Persiapan Data (Data Preparation)

Pada tahap ini, perlu mencari dan mengumpulkan semua *sample* gambar terkait objek batubara. Mulai dari percikan api, bara api, hingga bentuk api itu sendiri. Setelah terkumpul, data diolah dan dipilah.

Data yang sudah terkumpul dan sudah dipilah, perlu dilabelkan terlebih dahulu sebelum fase Pelatihan Data. Proses *labeling* dilakukan di Roboflow *workspace* sariawan27 dengan mode *traditional labeling*. Setelah semua data sudah diberi label, dataset tersebut bisa langsung di *export* di fitur *versions* pada Roboflow. Hasil export tersebut bisa langsung di *import* pada Google Collab untuk melakukan fase Pelatihan Data.

b. Fase Pelatihan Data (Training Model di Google Collab)

Pelatihan data/*training model* dilakukan menggunakan arsitektur YOLOv8 pada platform Google Collab. Pelatihan ini dilakukan sebanyak 50 *epochs* dengan ukuran gambar (*imgsz*) sebesar 640 piksel.

Proses pelatihan model ini membutuhkan ribuan matriks simultan

yang hanya bisa dilakukan secara cepat menggunakan GPU. Maka dari itu, penelitian ini menggunakan GPU NVIDIA TESLA T4 pada Google Collab untuk memproses matriks tersebut secara cepat.

Jika proses pelatihan sudah selesai, Google Collab akan menghasilkan *file best.pt*. Lalu *file best.pt* di *export/download* untuk digunakan pada fase implementasi.

c. Fase Implementasi dan Pengujian (Interface di Macbook)

Pada fase ini, *file best.pt* dipindahkan ke local (perangkat Macbook Pro 2017, CPU) yang nantinya digunakan pada *project* sistem deteksi yang menggunakan python 3.11. Tahapan pada fase ini meliputi:

- **Preprocessing:** *Frame* video dikonversi ke ruang warna HSV untuk membantu validasi karakter warna api.
- **Tracking:** Menggunakan algoritma ByteTrack agar menjaga konsistensi ID objek untuk setiap frame nya.
- **Post-processing:** Analisis arah gerak dan perubahan luas area api untuk mendeteksi potensi bahaya api.
- **Alerting System:** Sistem peringatan dini menggunakan notifikasi bot Telegram untuk mengirim status perbesaran api dan arah rambat api.

3.5. Parameter Pengujian

Untuk mengevaluasi kinerja model dan sistem, perlu memerhatikan parameter pengujian berikut:

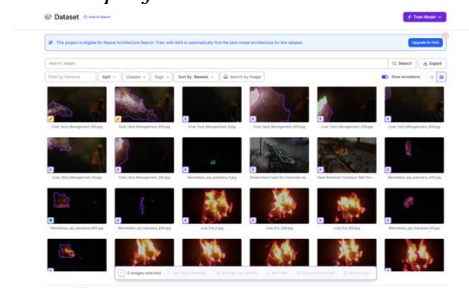
- Performa sistem:** Mengukur kecepatan deteksi (FPS) dan waktu respon sistem dalam deteksi objek.
- Akurasi arah pergerakan:** Mengukur akurasi dalam mendeteksi perpindahan objek.

- Pengujian Notifikasi:** Mengukur delay serta tingkat keberhasilan saat pengiriman notifikasi Telegram.

4. HASIL DAN PEMBAHASAN

Pada penelitian ini, penulis akan menjabarkan hasil eksperimen dan analisis mengenai performa deteksi api pada *belt conveyor*. Penulis mengimplementasikan algoritma YOLOv8 sebagai pengolahan citra untuk mengidentifikasi objek api, bara api, dan percikan api secara realtime. Tujuan dari pembahasan ini tidak hanya untuk menampilkan data mentah saja, namun juga membedah hubungan kausalitas antara algoritma YOLO dan optimalisasi kode program karena berpengaruh terhadap parameter pengujian.

Proses pemodelan diawali dengan pengumpulan *sample* gambar, lalu dijadikan satu folder *project* di Roboflow.



Gambar 2 Dataset Roboflow

Gambar 2 adalah folder yang berisikan *sample* gambar api, percikan api, bara api di *belt conveyor*. Lalu gambar-gambar itu diberi label dan jadilah dataset siap pakai. Karena untuk meminimalkan waktu komputasi dalam proses *training* dataset, maka proses tersebut dilakukan di Google Colab.

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="5MsGKKvDPFMTfAgZnX")
project = rf.workspace("sariawan27").project("prototype-coal-flame-detection")
version = project.version(1)
dataset = version.download("yolo8")
```

Gambar 3 Import Dataset

Pada gambar 3, dataset di *import* ke dalam google colabs guna dilakukannya *training data*. Karena dataset sudah di *import*, maka kita gunakan algoritma YOLO dari *ultralytics* seperti gambar 4 berikut ini.

```

from ultralytics import YOLO

model = YOLO("yolov8n.pt")

model.train(
    data="/content/Prototype-Coal-Flame-Detection/data.yaml",
    epochs=50,
    imgsz=640
)

```

Gambar 4 Train Data Using YOLO

Gambar 4 menunjukkan bahwa *epochs* yang diberikan adalah 50. *Epochs* tersebut digunakan untuk menentukan bahwa seluruh dataset akan dilewati jaringan saraf tiruan sebanyak 50 kali, baik secara *forward* maupun *backward*. Jumlah iterasi ini dipilih sebagai titik optimal untuk memberikan waktu *loss function* agar mengecil mendekati nilai konvergen, sekaligus mencegah kerusakan dalam akurasi pengujian di lapangan. Pada gambar 4 juga menunjukkan bahwa *imgsz* yang diberikan adalah 640. Nilai tersebut memaksa algoritma untuk *resizing* semua matriks citra menjadi resolusi 640x640 piksel sebelum memasuki *convulation layers*. Skala ini dipilih karena memiliki keseimbangan terbaik dalam mempertahankan detail spasial objek kecil sekaligus menjaga beban komputasi grafis tetap efisien.

Setelah proses *train dataset* selesai, hasil *export(best.pt)* dari Google Colabs diimport ke dalam kode program *python*. Perlu digaris bawahi bahwa kode program *python* yang dirancang dalam sistem ini bersifat tidak statis, melainkan terdapat beberapa kustomisasi logika algoritma pada *pipeline data*.

4.1. Kecepatan Pemrosesan (Inference Speed)

Karena *Macbook Pro 2017 intel i7* tidak bisa mengelola GPU maka pengujian pemrosesan diletakkan pada sisi *cloud* (Google Colab).

Dari riwayat di pelatihan data Google Colab, rata-rata waktu yang dibutuhkan untuk memproses satu *frame* sebagai berikut:

- a. **Preprocess:** 0.3 ms.
- b. **Inferences:** 4.06 ms.
- c. **Postprocess:** 2.92 ms.
- d. **Total waktu per frame:** ± 7.29 ms

Dengan total waktu tersebut bisa dibayangkan berjalan dengan cepat jika dijalankan dari sisi server (Google Collabs). Namun saat implementasi ke

local pasti akan menyesuaikan beban kinerja prosesor intel i7.

4.2. Akurasi Arah Gerak

Setelah objek api dideteksi oleh YOLOv8, lalu diberikan *track ID* guna analisis pergerakan objek. Analisis ini bertujuan untuk mengetahui posisi api sehingga dapat diketahui bahwa api bergerak ke arah mana.

```

# =====
# TRAJECTORY HISTORY
# =====
if track_id not in track_history:
    track_history[track_id] = []

track_history[track_id].append((int(x), int(y)))

```

Gambar 5 Track History

Proses pada gambar 5 memanfaatkan koordinat titik pusat dari *bounding box* hasil deteksi. Koordinat tersebut disimpan dalam *variable track_history* berdasarkan *track_id* masing-masing objek. Dengan demikian, sistem memiliki riwayat posisi objek sebelum hingga saat ini.

Untuk menentukan arah gerak, sistem perlu membagi citra menjadi beberapa zona sesuai kebutuhan. Pada contoh gambar 6 berikut, zona dibagi menjadi 4 bagian.

```

# =====
# 3. FUNGSI ZONA (INI INTI TAMBAHAN 🍷)
# =====
def get_zone(x, y, frame_w, frame_h):
    if x < frame_w / 2 and y > frame_h / 2:
        return "A" # kiri bawah
    elif x >= frame_w / 2 and y > frame_h / 2:
        return "B" # kanan bawah
    elif x >= frame_w / 2 and y <= frame_h / 2:
        return "D" # kanan atas
    else:
        return "C" # kiri atas

```

Gambar 6 Pembagian Zona

Setelah sistem membagi citra menjadi beberapa bagian, sistem perlu mendeteksi arah gerak. Arah gerak ditentukan berdasarkan perubahan koordinat sumbu (x) dan (y). Kondisional/algoritma arah gerak tersebut dapat dilihat dalam gambar 7 berikut.

```
# =====
# LOGIKA ARAH GERAK
# =====
arah = ""
if x > prev_x + 7: arah += "Kanan "
if x < prev_x - 7: arah += "Kiri "
if y < prev_y - 7: arah += "Atas "
if y > prev_y + 7: arah += "Bawah "
arah_final = arah.strip() if arah else "Diam"
```

Gambar 7 Kondisional arah gerak

Dari beberapa potong kode diatas, penulis melakukan pengujian manual dengan rumus berikut:

$$\text{Akurasi} = \frac{\text{Jumlah Prediksi Benar}}{\text{Total Data}} \times 100\% \quad (1)$$

Rumus di atas berfungsi untuk menghitung tingkat akurasi dari pengujian arah gerak objek. Berikut hasil ujinya:

Table 1 Hasil Uji Akurasi Arah Gerak

No.	Arah Sebenarnya	Hasil Sistem	Status
1.	Titik A - Titik B	Titik A - Titik B	Benar
2.	Titik B - Titik D	Titik B - Titik D	Benar
3.	Titik D - Titik C	Titik D - Titik C	Benar
4.	Titik C - Titik A	Titik C - Titik A	Benar

Dengan data uji sesuai table 1, akan diperoleh hasil akurasi sebagai berikut:

$$\text{Akurasi} = \frac{4}{4} \times 100\% = 100\%$$

4.3. Notifikasi

Sebagai bentuk peringatan dini (Early Warning System), sistem ini mengintegrasikan dengan notifikasi Telegram. Notifikasi akan dikirim Ketika sistem mendeteksi adanya objek api yang memenuhi kondisi tertentu seperti perpindahan zona atau perubahan bentuk objek.

Proses pengiriman notifikasi berada di bagian fungsi `send_telegram_alert` dengan beberapa parameter seperti yang ada di gambar 8 berikut ini.

```
INTERVAL_NOTIF = 15 # detik
def send_telegram_alert(frame, status_api, arah_gerak, track_id, zona):
```

Gambar 8 Fungsi Send Telegram

Di dalam fungsi pada gambar 8 terdapat beberapa kode yang mengonfigurasi sistem agar terhubung ke Telegram BOT API. Konfigurasi saat ini dapat dilihat pada gambar 9 berikut ini.

```
# --- PERBAIKAN URL MANUAL (PASTIKAN ADA /bot SEBELUM TOKEN) ---
api_url = f"https://api.telegram.org/bot{TOKEN}"

try:
    # 1. Kirim Pesan Teks
    r_text = requests.post(f"{api_url}/sendMessage", params={
        'chat_id': CHAT_ID,
        'text': pesan,
        'parse_mode': 'Markdown'
    }, timeout=5)

    # 2. Kirim Foto
    with open(img_name, 'rb') as f:
        requests.post(f"{api_url}/sendPhoto",
            data={'chat_id': CHAT_ID,
                'files': ('photo', f)},
            timeout=10)

    print(f"✅ Notifikasi Terkirim ke Telegram!")
    last_notif_time = current_time
except Exception as e:
    print(f"❌ Detail Error: {e}")
    print(f"❗ Pastikan TOKEN dan CHAT_ID sudah benar.")
```

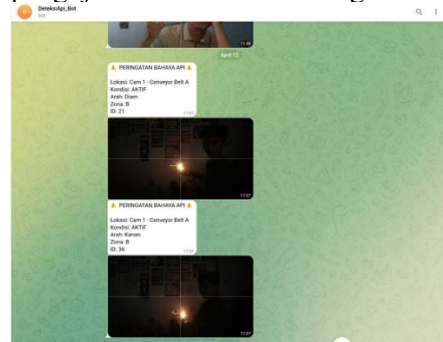
Gambar 9 Konfigurasi Telegram BOT API

Untuk konfigurasi pesan, ada beberapa poin utama yang harus ditampilkan. Poin tersebut meliputi: Lokasi, zona, arah gerak, kondisi. Konfigurasi dapat dilihat pada gambar 10 berikut ini.

```
pesan = (
    "⚠️ *PERINGATAN BAHAYA API* ⚠️\n\n"
    f"📍 *Lokasi:* {LOKASI_KAMERA}\n"
    f"🔥 *Kondisi:* {status_api}\n"
    f"👉 *Arah Gerak:* {arah_gerak}\n"
    f"📍 *Zona:* {zona}\n"
    f"📄 *ID Objek:* {track_id}\n\n"
    "🚨 *Segera lakukan pengecekan lokasi!*"
)
```

Gambar 10 Konfigurasi Pesan Telegram

Dari beberapa kode di atas perihal notifikasi, penulis melakukan pengujian terkait *Early Warning System*. Untuk hasil pengujian bisa dilihat dalam gambar 11.



Gambar 11 Pengujian notifikasi telegram

Gambar 11 membuktikan bahwa notifikasi telegram sudah terkirim lengkap dengan informasi lokasi kamera, zona/titik api dan status api. Dari gambar 11 juga menunjukkan bahwa *delay* notifikasi telegram kurang dari 1 menit dalam 2 kali mendeteksi api.

5. KESIMPULAN

Berdasarkan hasil pengujian performa model api, bara api, dan percikan api dari dataset Roboflow *workspace sariawan27 project prototype-coal-flame-detection*, dapat disimpulkan bahwa model memiliki kecepatan pemrosesan yang bisa dibilang cepat dengan total waktu per-*frame*-nya ± 7.29 ms. Pengujian akurasi arah gerak api juga menunjukkan nilai 100%. Selain itu, pengujian notifikasi telegram juga memiliki *delay* yang minimum, yaitu 2 kali deteksi api hanya dengan waktu kurang dari 1 menit. Dengan demikian, sistem mampu membantu petugas untuk mempermudah proses penanganan risiko bahaya dan evakuasi karena performa sistem yang cepat, akurat, dan notifikasi yang real-time. Untuk kedepannya, disarankan agar dilakukan pengembangan lebih lanjut terhadap model yang ada di dataset Roboflow *workspace sariawan27 project prototype-coal-flame-detection*.

DAFTAR PUSTAKA

- [1] M. Sohib and G. M. Kusbiantoro, "Perencanaan Belt Conveyor Batu Bara dengan Kapasitas 1000 Ton per Jam di PT. Meratus Jaya Iron Steel Tanah Bumbu," *Wahana Teknik*, vol. 7, no. 1, 2018.
- [2] L. Wang *et al.*, "Review of conveyor belt fires in coal mines: A multiscale analysis of combustion, smoke propagation, and intelligent monitoring," *Green and Smart Mining Engineering*, vol. 3, no. 1, pp. 16–26, 2026, doi: 10.1016/j.gsme.2025.12.001.
- [3] M. N. Abdurrahman, "Belt Conveyor PLTU Jeneponto Sulsel Terbakar," *detikNews*, Detikcom, 2017. [Online]. Available: <https://news.detik.com/berita/d-3653046/belt-conveyor-pltu-jeneponto-sulsel-terbakar>
- [4] G. Times, "Coal mine fire disaster in Southwest China's Guizhou kills 16," *Global Times*, Global Times, 2023. [Online]. Available: <https://www.globaltimes.cn/page/202309/1298819.shtml>
- [5] A. Kumar and J. Maya, "Advanced Fire Detection: Leveraging Deep Learning and Heat Signature Analysis for Enhanced Accuracy," *Journal of Frontiers in Multidisciplinary Research*, vol. 6, no. 1, pp. 14–29, 2025, doi: 10.54660/IJFMR.2025.6.1.14-29.
- [6] B. Bestari and M. I. Zulfa, "Deteksi Kerusakan Conveyor Belt Real Time Berbasis YOLOv11 EdgeTPU pada Raspberry Pi 5," *Jurnal SINTA: Sistem Informasi dan Teknologi Komputasi*, vol. 3, no. 1, pp. 7–15, 2025, doi: 10.61124/sinta.v3i1.138.
- [7] R. Leonardo, "Perbandingan Kinerja dan Efisiensi Arsitektur CNN VGG-16 dan ResNet50 dalam Kanker Kulit Dataset HAM10000," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 14, no. 2, 2026, doi: 10.23960/jitet.v14i2.9351.
- [8] T. A. Dompeipen and S. R. U. A. Sompie, "Penerapan Computer Vision Untuk Pendeteksian Dan Penghitungan Jumlah Manusia," *Jurnal SINTA: Sistem Informasi dan Teknologi Komputasi*, 2021.
- [9] E. Arkin, N. Yadikar, X. Xu, A. Aysa, and K. Ubul, "A survey: object detection methods from CNN to transformer," *Multimed. Tools Appl.*, vol. 82, no. 14, pp. 21353–21383, 2023, doi: 10.1007/s11042-022-13801-3.
- [10] A. Bouguettaya, H. Zarzour, A. M. Taberkit, and A. Kechida, "A review on early wildfire detection from unmanned aerial vehicles using deep learning-based computer vision algorithms," *Signal Processing*, vol. 190, p. 108309, 2022, doi: 10.1016/j.sigpro.2021.108309.
- [11] A. Arroyo, R. Fernandez-Caballero, and F. J. Martinez, "Deep Learning-Based Fire and Smoke Detection in Open Areas," *Sensors*, vol. 21, no. 9, 2021.
- [12] A. E. Cetin and others, "Video Fire and Smoke Detection Using Deep Learning," *IEEE Access*, 2021.
- [13] Ultralytics, "YOLOv8 Documentation," 2023.
- [14] G. Jocher and others, "YOLOv7 vs YOLOv8 Performance Comparison," *ArXiv*, 2023.
- [15] G. A. Suni and L. K. Balol, "Penerapan Teknologi IoT untuk Sistem Peringatan Dini Gempa dengan Notifikasi Telegram dan Blynk," *Jurnal Komputer dan Teknologi*, vol. 5, no. 1, 2025, doi: 10.64626/jukomtek.v5i1.540.
- [16] N. D. H. Manurung, R. Khana, J. Muliadi, and M. Sobirin, "SISTEM DETEKSI KEBAKARAN BERBASIS VISI KOMPUTER DENGAN YOLOv11 TERINTEGRASI DENGAN TELEGRAM," *Transmisi: Jurnal Ilmiah Teknik Elektro*, vol.

27, no. 4, pp. 240–248, Apr. 2025, doi:
10.14710/transmisi.27.4.240-248.