

Revisi:

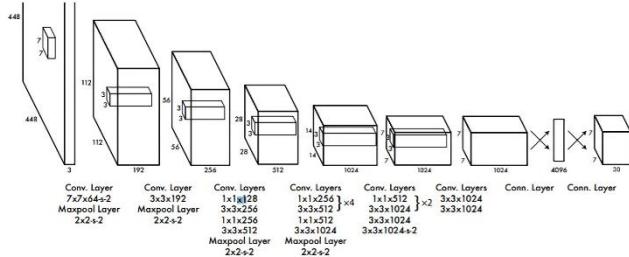


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Secara keseluruhan, gambar ini mengonfirmasi bahwa arsitektur "YOLOv11" *Saya* mengikuti standar arsitektur YOLO modern yang terdiri dari tiga bagian utama. Mari kita urutkan alirannya:

1. Input Citra (*Image Input*)

Keterangan: Input: 640x640

Artinya: Sistem *Saya* dirancang untuk menerima gambar masukan dengan resolusi 640x640 piksel. Ukuran ini merupakan kompromi yang baik antara kecepatan komputasi dan kemampuan model untuk mendekksi objek dengan detail yang cukup. Ini sesuai dengan parameter imgsz=640 pada notebook pelatihan *Saya*.

2. Backbone: CSPDarknet53

Keterangan: *Backbone*: CSPDarknet53

Artinya: Ini adalah "tulang punggung" atau pengekstraksi fitur utama dari model *Saya*.

- a) CSPDarknet53 adalah arsitektur *Backbone* yang sangat efisien dan kuat. Nama ini merupakan gabungan dari CSP (*Cross Stage Partial*) dan Darknet53.
- b) Fungsi Utamanya: Seperti yang terlihat dari panah-panah yang mengalir ke bawah, *Backbone* ini mengambil gambar 640x640 dan secara progresif mengekstraksi fitur-fitur visual sambil mengurangi dimensi spasial gambar (dari 640x640 menjadi 20x20).
- c) Konteks Penting: Arsitektur *Backbone* CSPDarknet53 adalah salah satu ciri khas utama dari YOLOv4 dan YOLOv5. Ini adalah petunjuk terkuat bahwa model "YOLOv11" *Saya* secara arsitektural sangat mirip atau berbasis pada YOLOv5.

3. Neck: SPPF & PANet

Keterangan: *Neck*: SPPF, PANet

Artinya: Ini adalah bagian "leher" yang bertugas menggabungkan fitur-fitur yang telah diekstraksi oleh *Backbone*.

- a) SPPF (Spatial Pyramid Pooling-Fast): Ini adalah sebuah modul yang berfungsi untuk menangkap informasi konteks dalam berbagai skala tanpa memperlambat proses secara signifikan. Ia membantu model untuk memahami objek yang sama dalam berbagai ukuran.
- b) PANet (Path Aggregation Network): Ini adalah mekanisme fusi fitur yang canggih. Seperti yang digambarkan oleh alur panah yang kompleks di bagian tengah:
 - Alur *Top-Down* (Atas ke Bawah): Membawa informasi semantik (konteks "apa" objeknya) dari fitur level tinggi ke fitur level rendah.
 - Alur *Bottom-Up* (Bawah ke Atas): Membawa informasi lokalisasi yang presisi dari fitur level rendah ke fitur level tinggi.
- c) Kombinasi ini menghasilkan feature map yang sangat kaya, memungkinkan model mendeteksi objek besar dan kecil secara bersamaan dengan sangat efektif. Sekali lagi, kombinasi SPPF dan PANet adalah ciri khas utama dari arsitektur YOLOv5.

4. Head (Prediction): YOLOv3 Head

Keterangan: *Head (Prediction)*: YOLOv3 Head

Artinya: Ini adalah bagian "kepala" yang melakukan prediksi akhir.

- a) Ia mengambil tiga feature map dari *Neck* pada skala yang berbeda (ditunjukkan oleh tiga panah yang keluar dari PANet). Setiap skala bertanggung jawab untuk mendeteksi objek dengan ukuran yang berbeda (misalnya, skala besar untuk objek kecil, skala sedang untuk objek medium, dan skala kecil untuk objek besar).
- b) Meskipun diberi label "*YOLOv3 Head*", fungsinya tetap sama: menghasilkan *Bounding Box*, *Confidence Score*, dan *Class Probability* untuk setiap objek yang terdeteksi.

Arsitektur Umum YOLO: Tiga Komponen Utama

Bayangkan YOLO sebagai sebuah "pabrik" pemrosesan gambar. Gambar masuk dari satu sisi, lalu diproses oleh tiga departemen utama sebelum menghasilkan output deteksi objek.

[Gambar Input] --> [Backbone] --> [Neck] --> [Head] --> [Output Deteksi]

Berikut penjelasan masing-masing "departemen":

1. Backbone (Tulang Punggung)

- a. **Tugas Utama:** Mengekstraksi fitur-fitur penting dari gambar.
- b. **Analogi:** Anggap *Backbone* seperti **mata dan korteks visual awal** pada manusia. Saat Saya melihat sebuah gambar, bagian ini bertugas untuk mengenali bentuk-bentuk dasar, garis, tekstur, dan warna tanpa langsung memberi nama pada objeknya.
- c. **Cara Kerja:** Terdiri dari jaringan syaraf konvolusi (*Convolutional Neural Network*) yang dalam dan padat. Ia mengambil gambar input (misalnya resolusi 640x640) dan secara bertahap mengecilkannya sambil "menebalkan" informasinya. Hasilnya adalah beberapa *feature map* (peta fitur) dalam berbagai skala yang menangkap informasi dari level rendah (garis, tepi) hingga level tinggi (bagian objek yang lebih kompleks).
- d. **Contoh:** CSPDarknet53 (digunakan di YOLOv4 & v5), E-ELAN (di YOLOv7).

2. Neck (Leher)

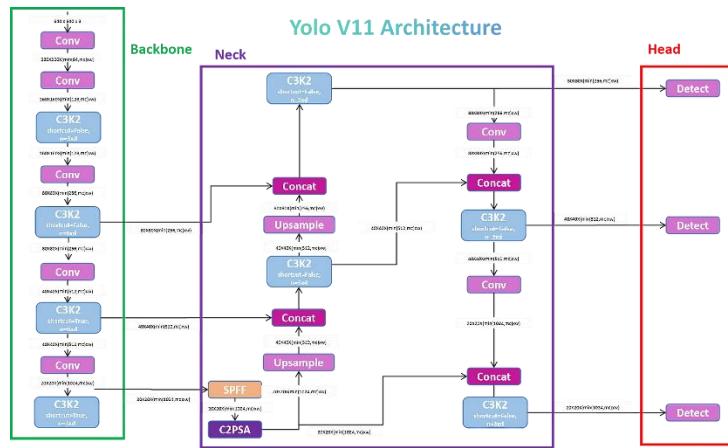
- a. **Tugas Utama:** Menggabungkan dan mencampur fitur dari berbagai skala.
- b. **Analogi:** *Neck* adalah **pusat asosiasi di otak**. Ia mengambil informasi visual dasar dari *Backbone* (misalnya, fitur objek besar dan fitur objek kecil) lalu menggabungkannya. Ini penting agar model bisa mendeteksi objek berukuran kecil sekaligus objek berukuran besar dalam satu gambar.
- c. **Cara Kerja:** Menggunakan mekanisme seperti *Feature Pyramid Network* (FPN) dan *Path Aggregation Network* (PANet). FPN mengambil fitur dari atas ke bawah (dari abstrak ke detail), sementara PANet menambahkan jalur dari bawah ke atas (dari detail ke abstrak). Proses "bolak-balik" ini menghasilkan *feature map* yang jauh lebih kaya informasi sebelum diserahkan ke bagian kepala.
- d. **Contoh:** PANet, BiFPN.

3. Head (Kepala)

- a. **Tugas Utama:** Melakukan prediksi akhir.
- b. **Analogi:** *Head* adalah **bagian otak yang memberi keputusan akhir**. Setelah menerima informasi yang kaya dari *Neck*, bagian ini akan berkata, "Oke, di koordinat ini ada objek, saya 89% yakin ini 'Api', dan di koordinat itu ada objek lain, saya 99% yakin ini 'Asap'."
- c. **Cara Kerja:** Ini adalah bagian detektornya. Ia bekerja pada *feature map* yang sudah diperkaya oleh *Neck* dan menghasilkan output akhir berupa:
 - a) **Bounding Box:** Koordinat (x, y, lebar, tinggi) dari kotak yang melingkupi objek.

- b) **Class Probability:** Probabilitas objek tersebut termasuk dalam kelas tertentu (misalnya, 89% Api, 11% Asap).
- c) **Confidence Score:** Tingkat kepercayaan model bahwa di dalam kotak tersebut memang ada sebuah objek.

2. Ringkasan Arsitektur YOLOv11



◆ 1. Backbone (Warna Hijau) – Ekstraksi Ciri-Ciri Visual

Bagian *backbone* adalah tempat pertama input gambar diproses. Tujuannya adalah mengambil informasi penting dari gambar, seperti bentuk, tekstur, atau pola yang mungkin menunjukkan keberadaan objek (dalam kasus kamu: api dan asap).

Komponen penting di backbone:

- Conv (*Convolution*): Lapisan ini mendeteksi pola visual dari gambar seperti tepi, warna, atau bentuk.
- C3K2 (*Cross-Stage Partial with 3 Conv Kernel Size 2*): Modul ini membantu menyimpan informasi penting sambil mempercepat proses. C3K2 adalah versi efisien dari modul C3 pada YOLOv5.
- Downsampling: Mengcilkan ukuran gambar agar sistem bisa melihat gambaran besar dan mempercepat proses. Setiap kali *downsampling*, detail makin sedikit tapi makna makin dalam.
- Shortcut connection: Penghubung dari lapisan awal ke lapisan selanjutnya agar tidak kehilangan informasi penting (*residual learning*).

Analogi: Bayangkan kamu memotret kebakaran. Di sini, sistem mulai mengenali warna merah menyala, bentuk kepulan asap, dan intensitas cahaya.

◆ 2. Neck (Warna Ungu) – Penggabungan dan Penguatan Fitur

Bagian *neck* bertugas menggabungkan informasi dari berbagai ukuran gambar. Ini penting karena objek bisa muncul besar (api besar), sedang, atau kecil (api kecil atau asap tipis).

Komponen penting di neck:

- Concat (*Concatenation*): Menyatukan dua atau lebih hasil dari lapisan sebelumnya agar fitur saling melengkapi.
- Upsample: Membesarkan kembali citra fitur agar bisa digabung dengan fitur dari lapisan sebelumnya.
- SPPF (*Spatial Pyramid Pooling - Fast*): Teknik ini menggabungkan informasi dari beberapa ukuran filter untuk memahami objek dengan berbagai skala secara efisien.
- C2PSA (*Cross-Channel Partial Spatial Attention*): Modul ini memberi perhatian lebih pada bagian gambar yang penting, seperti area yang menyala atau memiliki kepulan asap, dan mengabaikan latar belakang yang tidak relevan.

Analogi: Ini seperti kamu melihat gambar dari berbagai sudut dan ukuran untuk memastikan benar-benar itu asap atau bukan bayangan.

● 3. Head (Warna Merah) – Prediksi Akhir (Deteksi Objek)

Bagian *head* adalah otak terakhir yang membuat keputusan: “Apakah ini api?” atau “Apakah ini asap?”, serta di mana letaknya di gambar.

Komponen penting di head:

- Detect layer: Terdiri dari 3 lapisan, masing-masing mendeteksi objek dari ukuran besar, sedang, hingga kecil.
 - Skala besar: Untuk objek besar seperti api besar.
 - Skala sedang: Untuk api biasa atau asap padat.
 - Skala kecil: Untuk api kecil atau asap samar.

Sistem ini menggunakan pendekatan *anchor-free*, artinya model langsung memprediksi pusat objek, ukurannya, dan kelasnya—tanpa perlu tebakan awal (*anchor box*).

Komponen	Fungsi	Perubahan Utama di YOLOv11
Backbone	Ekstraksi fitur dasar (tepi, tekstur, warna)	Menggunakan <i>CSPDarknet</i> yang lebih ringan + <i>Transformer block</i> kecil untuk konteks global
Neck (FPN+PAN)	Menggabungkan fitur multi-skala agar objek kecil & besar terdeteksi sama baik	<i>Lightweight PAN++</i> mempercepat inferensi
Head (Detection)	Memprediksi <i>Bounding Box, class, confidence</i>	<i>Anchor-free</i> : tidak pakai <i>anchor boxes</i> → lebih sedikit parameter & lebih akurat untuk objek kecil
Output Akhir	3 skala prediksi (<i>small, medium, large</i>)	Cocok untuk api/asap yang ukurannya bervariasi

Sederhananya: citra → *Backbone* → *Neck* → *Head* → kotak deteksi + label “fire” atau “smoke”.

1. ***Backbone – Feature Extractor***

Fungsi: mengekstrak representasi fitur dari citra input.

Blok utama:

- Conv Block → 2-D Convolution + BatchNorm + SiLU (aktivasi yang lebih smooth dari ReLU).
- BottleNeck → residual block mirip ResNet; bisa dipilih untuk memakai shortcut atau tidak.
- C3K2 (Cross-Stage Partial Kernel-2) → inti baru YOLOv11. C3K2 menggantikan C2f di YOLOv8. Ia memecah map fitur, menjalankan beberapa 3×3 convolution yang lebih murah komputasi, lalu menggabungkan kembali sehingga parameter lebih sedikit tetapi representasi lebih kaya.

2. ***Neck – Multi-Scale Fusion***

Fungsi: menyatukan fitur dari berbagai tingkat (skala) agar model mampu mendeteksi objek kecil hingga besar.

Komponen kunci:

- SPFF (*Spatial Pyramid Pooling – Fast*) → kumpulan max-pooling dengan ukuran kernel berbeda untuk menangkap konteks multi-skala tanpa menambah resolusi feature map .
- Upsample + PANet (*Path Aggregation Network*) → jalur dua arah (*Top-Down & Bottom-Up*) memastikan informasi geometri tingkat rendah dan semantik tingkat tinggi tergabung.

3. ***Head – Anchor-Free Detection***

Fungsi: menghasilkan *Bounding Box*, *class label*, dan *Confidence Score*.

- Anchor-free → memprediksi langsung titik pusat objek tanpa anchor box → mengurangi jumlah parameter dan lebih adaptif pada objek kecil .
- C2PSA (*Cross Stage Partial with Spatial Attention*) → dua cabang Partial-Spatial-Attention (PSA) dipakai untuk menekankan region penting (mis. asap samar) sebelum hasil akhir dikeluarkan .

Ringkasan perbedaan YOLOv11 vs pendahulu:

- C3K2 lebih ringan & cepat dari C2f.
- SPFF + C2PSA meningkatkan deteksi objek kecil.
- *Head anchor-free* memangkas kompleksitas.

Dengan susunan ini, YOLOv11 mampu menjaga kecepatan real-time sekaligus meningkatkan akurasi, terbukti di skripsi Saya dengan $mAP@0.5 = 0,88$ dan F1-Score Fire 99,52 %.

3. Mengapa *F1-Score* Dipakai & Perbandingan Metrik

3.1 Definisi Singkat

- Precision** = $TP / (TP + FP)$
- Recall** = $TP / (TP + FN)$
- F1-Score** = $2 \times (Precision \times Recall) / (Precision + Recall)$

3.2 Alasan *F1-Score*

- Seimbang: menggabungkan *Precision* (kebenaran alarm) & *Recall* (kemampuan menangkap semua kebakaran).
- Cocok untuk dataset tidak seimbang: kasus kebakaran lebih sedikit dibanding latar belakang.
- Mudah diinterpretasi: satu angka 0–1, makin tinggi makin baik.

3.3 Perbandingan Metrik Lain

Metrik	Kelebihan	Kekurangan	Kapan Dipakai
Accuracy	Mudah dipahami	Tidak akurat jika data tidak seimbang	Deteksi spam e-mail
mAP@0.5	StSaya kompetisi <i>object Detection</i>	Banyak ambang IoU → sulit interpretasi cepat	Benchmark COCO
Precision/Recall terpisah	Lihat <i>trade-off</i> jelas	Dua angka → perlu visualisasi kurva	Analisis lanjutan
F1-Score	Ringkas & seimbang	Tidak memperlihatkan trade-off	Laporan akhir & keputusan cepat

Contoh: sistem memiliki *Precision* 0.89 & *Recall* 0.87 → $F1 = 0.88$. Angka ini langsung memberi gambaran “secara keseluruhan, 88 % deteksi benar-benar kebakaran dan tidak ada yang terlewat”.

Ringkasan untuk Bab

“Lapisan konvolusi pada YOLOv11 bekerja dengan cara menggeser *kernel* 3×3 di atas citra, menghitung dot-product, lalu mengaktivasi hasilnya dengan ReLU. Proses ini diulang berlapis-lapis di *Backbone* hingga menghasilkan *feature maps* yang kaya informasi. Hasil akhir dari *Detection Head* berupa kotak pembatas di sekitar api/asap dengan nilai *confidence* tinggi. Untuk mengevaluasi performa, *F1-Score* dipilih karena mencerminkan keseimbangan antara alarm palsu (*Precision*) dan kebakaran yang terlewat (*Recall*) pada dataset yang tidak seimbang.”

Pertanyaan 1: "Coba jelaskan secara singkat, apa yang melatarbelakangi Saya memilih judul penelitian ini? Apa urgensinya?"

Penelitian ini dilatarbelakangi oleh meningkatnya frekuensi dan risiko bencana kebakaran di Indonesia. Sistem deteksi kebakaran konvensional seringkali memiliki keterbatasan infrastruktur dan biaya implementasi yang tinggi, terutama dalam hal pemasangan sensor fisik. Oleh karena itu, penelitian ini bertujuan mengembangkan solusi alternatif yang lebih ekonomis, fleksibel, dan mudah diakses dengan memanfaatkan kemajuan teknologi Kecerdasan Buatan (AI), khususnya. *Computer Vision*. Urgensinya terletak pada kebutuhan akan sistem peringatan dini (*early warning*) yang cepat dan akurat untuk memitigasi dampak luas kebakaran, baik dari segi kerugian materi, keselamatan jiwa, maupun kerusakan lingkungan.

Pertanyaan 2: Mengapa Saya secara spesifik memilih algoritma YOLOv11 dan *framework* Streamlit? Apa keunggulan utama keduanya untuk kasus deteksi api dan asap?

1. YOLOv11: Algoritma *You Only Look Once* (YOLO) versi 11 dipilih karena keunggulannya dalam deteksi objek secara *real-time* dengan tingkat akurasi tinggi dan latensi yang rendah. Sebagai versi terbaru, YOLOv11 menawarkan peningkatan efisiensi dan akurasi dibandingkan versi-versi sebelumnya, yang sangat krusial untuk deteksi dini kebakaran di mana kecepatan adalah faktor penentu.
2. Streamlit: *Framework* Streamlit dipilih karena kemudahannya dalam integrasi dengan model *Deep Learning* dan kemampuannya untuk membangun antarmuka web yang interaktif dengan cepat tanpa memerlukan infrastruktur *cloud* yang kompleks. Hal ini memungkinkan sistem menjadi ringan, mudah diakses melalui web, dan dapat dijalankan pada server ringan, sehingga menekan biaya dan kompleksitas *deployment*.

Pertanyaan 3: "Saya menggunakan metodologi CRISP-DM. Bisa jelaskan tahapan-tahapan yang Saya lakukan dalam penelitian ini sesuai kerangka kerja tersebut?"

Penelitian ini mengadopsi metodologi CRISP-DM yang terdiri dari beberapa tahapan sistematis:

1. *Business Understanding*: Tahap awal di mana saya mengidentifikasi masalah utama, yaitu kebutuhan akan sistem deteksi kebakaran yang efisien, dan menetapkan tujuan proyek untuk membangun sistem deteksi berbasis Computer Vision.

2. Data Understanding: Pada tahap ini, saya melakukan eksplorasi terhadap dataset yang akan digunakan, yaitu "Smoke and Fire v2" dari Roboflow. Saya mempelajari karakteristik data, seperti jumlah gambar (9.848 citra), jumlah kelas (Api dan Asap), serta distribusinya.
3. Data Preparation: Tahap ini mencakup proses persiapan data sebelum pelatihan model. Saya melakukan labelling, augmentasi data (seperti mosaic, rotasi, dan penyesuaian kecerahan) untuk meningkatkan keragaman data, serta pembagian data (split data) menjadi data latih (8.243 gambar), validasi (1.062 gambar), dan uji (543 gambar).
4. Modelling: Di tahap ini, saya melakukan proses pelatihan model YOLOv11 menggunakan data yang telah disiapkan. Pelatihan dilakukan di Google Colaboratory selama 30 epoch.
5. Evaluation: Setelah model dilatih, saya melakukan evaluasi performa menggunakan metrik kuantitatif seperti Confusion Matrix, mAP@0.5, precision, recall, dan *F1-Score* untuk mengukur akurasi dan kesayalan model.
6. Deployment: Tahap akhir di mana model yang telah dievaluasi diintegrasikan ke dalam antarmuka web menggunakan Streamlit dan dilengkapi notifikasi Telegram, sehingga menjadi aplikasi yang siap pakai.

Pertanyaan 4: "Jelaskan secara konseptual, bagaimana cara kerja algoritma YOLO dalam mendekripsi objek?"

YOLO (You Only Look Once) bekerja dengan pendekatan yang berbeda dari detektor objek dua tahap seperti R-CNN. Konsep dasarnya adalah memperlakukan deteksi objek sebagai masalah regresi tunggal. Pertama, YOLO membagi gambar masukan menjadi sebuah grid berukuran $S \times S$. Setiap sel dalam grid ini bertanggung jawab untuk mendekripsi objek yang pusatnya jatuh di dalam sel tersebut. Setiap sel akan memprediksi beberapa *Bounding Box* beserta *Confidence Score* untuk setiap kotak tersebut, dan juga probabilitas kelas objek (*Class Probability*).

Confidence Score mencerminkan seberapa yakin model bahwa kotak tersebut berisi objek dan seberapa akurat prediksinya. Dengan memproses seluruh gambar hanya dalam satu kali inferensi, YOLO dapat mencapai kecepatan deteksi yang sangat tinggi dan cocok untuk aplikasi real-time.

Pertanyaan 5: "Dalam *notebook* Saya, terlihat proses pelatihan dilakukan selama 30 *epoch*. Apa dasar pertimbangan Saya dalam memilih jumlah *epoch* tersebut? Dan bagaimana Saya memonitor agar tidak terjadi *overfitting*?"

Jumlah 30 *epoch* dipilih sebagai titik awal yang umum untuk mencapai konvergensi pada *dataset* dengan ukuran seperti ini tanpa memakan waktu komputasi yang berlebihan. Berdasarkan *log* pelatihan dari *notebook*, dapat dilihat bahwa metrik evaluasi seperti mAP@0.5 dan *loss* pada data validasi menunjukkan peningkatan yang stabil dan mulai konvergen mendekati *epoch* ke-30. Untuk memonitor *overfitting*, saya memantau nilai *loss* pada data latih dan data validasi di setiap *epoch*. *Overfitting* ditunjukkan jika *loss* pada data latih terus menurun sementara *loss* pada data validasi mulai meningkat. Selain itu, penggunaan teknik augmentasi data seperti *mosaic*, rotasi, dan penyesuaian kecerahan juga secara signifikan membantu mengurangi risiko *overfitting* dengan memperkaya variasi data latih.

Pertanyaan 6: "Dari hasil evaluasi, model Saya mencapai mAP@0.5 sebesar 0.88. Jelaskan apa arti metrik ini dan bagaimana Saya menginterpretasikan nilai tersebut dalam konteks penelitian Saya."

mAP@0.5 atau *Mean Average Precision* pada ambang batas IoU (*Intersection over Union*) 0.5 adalah metrik yang digunakan untuk mengevaluasi performa model deteksi objek. Nilai ini merepresentasikan rata-rata dari *Average Precision* (AP) untuk semua kelas objek (dalam hal ini Api dan Asap). AP sendiri dihitung dari kurva *Precision-Recall*. Nilai 0.88 menunjukkan bahwa model memiliki performa yang sangat baik dalam mendekripsi lokasi dan mengklasifikasikan objek api dan asap dengan benar. Ini berarti, secara rata-rata, model memiliki kombinasi presisi dan *recall* yang tinggi, yang mengindikasikan keandalannya sebagai sistem deteksi.

Pertanyaan 7: "Pada *notebook* dan skripsi, Saya menampilkan *Confusion Matrix* dan *Classification Report*. Apa temuan utama yang bisa Saya simpulkan dari kedua hasil tersebut?"

Dari Confusion Matrix:

Confusion matrix menunjukkan bahwa model sangat jarang melakukan kesalahan klasifikasi antara kelas Api dan Asap. Jumlah *false positives* dan *false negatives* sangat rendah. Sebagai contoh, dari 726 sampel api pada data uji, hanya 4 yang salah didekripsi sebagai asap, dan

dari 263 sampel asap, hanya 3 yang salah dideteksi sebagai api. Ini menunjukkan model memiliki daya diskriminatif yang tinggi.

Dari Classification Report:

Laporan ini memberikan rincian performa per kelas. Kelas "Api" mencapai *F1-Score* 99.52% dan kelas "Asap" 98.67%. Nilai *F1-Score* yang tinggi untuk kedua kelas ini mengonfirmasi bahwa model memiliki keseimbangan yang sangat baik antara *precision* dan *recall*, artinya model tidak hanya akurat ketika melakukan prediksi tetapi juga sensitif dalam menemukan sebagian besar objek yang relevan di dalam gambar.