PENGOLAHAN CITRA · MATERI 12

YOLO Object Detection pada Video dan Webcam

Pelajari konsep melalui contoh yang dekat dengan keseharian, coba ilustrasi dan foto sendiri, lalu jalankan praktikum secara bertahap di Google Colab.

Pengantar → konsep → ilustrasi dan upload foto → video → praktikum

1. Mulai dari gambaran sederhana

Bayangkan sebuah kamera mengarah ke jalan. Pada satu foto terlihat sebuah bus. Sesaat kemudian, kamera menghasilkan foto berikutnya dan bus itu masih berada di sana, hanya posisinya sedikit bergeser. Jika YOLO mendeteksi bus pada kedua foto, program menghasilkan dua kemunculan deteksi. Namun, belum tentu ada dua bus berbeda. Perbedaan antara kemunculan di layar dan benda yang benar-benar berbeda menjadi dasar penting ketika deteksi gambar diterapkan pada video.

Video dapat dipahami sebagai rangkaian gambar yang memiliki urutan waktu. Model mendeteksi objek pada sebuah frame, lalu proses yang sama diulang untuk frame berikutnya. Walaupun terlihat sederhana, program juga perlu membaca berkas, memeriksa informasi video, menyesuaikan ukuran gambar, menggambar kotak, mencatat hasil, dan menulis video keluaran. Dengan demikian, kecepatan model hanyalah salah satu bagian dari kecepatan keseluruhan sistem.

Bab ini menghubungkan hasil deteksi dengan waktu. Kita akan membedakan laju pemutaran video dari laju komputasi, membaca perubahan jumlah kotak sepanjang klip, dan memahami mengapa tabel deteksi belum memberikan identitas objek. Kita juga melihat perbedaan antara membaca file video dan mengambil gambar dari webcam pengguna ketika Python berjalan di Google Colab, bukan di laptop pengguna.

Ilustrasi foto interaktif membantu memperhatikan hubungan antara satu tampilan dan deteksi di dalamnya; ilustrasi itu bukan bukti bahwa identitas objek sudah dilacak. Video pembelajaran kemudian memperluas gambaran tentang deteksi dan tracking. Setelah konsepnya jelas, praktikum Colab menjalankan pemrosesan berkas yang sebenarnya sehingga video anotasi, tabel, serta laporan waktunya berasal dari input dan komputasi nyata, bukan dari simulasi pada halaman web.

2. Pahami konsep secara berurutan

2.1 Dari satu gambar menjadi rangkaian frame

Pada deteksi gambar, masukan model adalah satu citra dan keluarannya berupa kotak, kelas, serta confidence. Pada video, bentuk pekerjaan dasarnya tetap sama, tetapi dilakukan berulang menurut urutan frame. Urutan ini penting karena perubahan posisi dan jumlah deteksi mempunyai makna temporal. Sebuah kotak yang muncul pada awal klip tidak otomatis berhubungan dengan kotak sekelas yang muncul menjelang akhir.

Praktikum membaca video melalui `cv2.VideoCapture`, kemudian mengambil dan memproses satu frame pada setiap putaran. Semua citra tidak ditumpuk di memori sekaligus. Cara ini lebih terkendali untuk klip yang panjang, walaupun tabel ringkasan tetap bertambah selama pemrosesan. Sebelum loop berjalan, program memeriksa apakah video terbuka serta apakah dimensi, FPS, jumlah frame, dan durasinya masuk batas. Ekstensi yang benar saja belum membuktikan isi video dapat dibaca.

2.2 Tiga laju yang menjawab pertanyaan berbeda

FPS sumber menjelaskan banyaknya frame yang dimainkan per detik menurut waktu video. Angka ini membantu menentukan timestamp dan laju pemutaran keluaran. Sebaliknya, **FPS inferensi** menyatakan banyaknya frame yang diproses dibandingkan total waktu pemanggilan model. Video dengan pemutaran lancar tidak menjamin model dapat mengikutinya pada CPU atau GPU yang sedang digunakan.

Throughput pipeline mencakup pekerjaan yang lebih luas: membuka sumber, membaca dan menyesuaikan frame, memanggil model, membuat anotasi, serta menulis keluaran sementara. Dalam kode, laporan `processing_fps_before_transcode` berhenti sebelum konversi video akhir. Karena itu, angka tersebut belum mencakup seluruh waktu sampai video tampil di notebook. Pengukuran memakai `perf_counter` dan sinkronisasi GPU; pemanggilan awal masih dapat memuat biaya pemanasan. Membandingkan kecepatan perlu menyebut perangkat, ukuran masukan, dan bagian proses yang benar-benar diukur.

2.3 Kotak berulang tidak sama dengan objek unik

Untuk setiap frame, program mencatat waktu video, jumlah kotak, waktu inferensi, dan jumlah deteksi setiap kelas. Grafiknya memperlihatkan perubahan deteksi sepanjang klip. Naik turunnya garis dapat berasal dari objek yang masuk atau keluar gambar, tetapi juga dari objek tertutup, gambar buram, maupun keputusan model yang berubah. Grafik bukan catatan pasti tentang jumlah benda fisik yang datang.

Laporan `sum_frame_detections_NOT_unique` menjumlahkan kemunculan kotak pada seluruh frame. Objek yang sama dapat menyumbang berkali-kali selama masih terlihat. Untuk

menghubungkan kemunculan tersebut, sistem memerlukan *tracking*, yaitu asosiasi antarframe yang memberi identitas track. Identitas pun dapat terputus saat objek menghilang atau tertukar ketika objek berdekatan. Penghitung kendaraan yang melewati garis membutuhkan aturan arah dan riwayat lintasan agar satu track tidak dihitung berulang. Kode bab ini menjalankan deteksi, bukan rangkaian tracking tersebut.

2.4 Ukuran gambar, koordinat, dan video keluaran

Sebelum inferensi, frame disesuaikan dengan skala yang sama pada kedua arah, lalu dimensinya dibulatkan menjadi genap untuk penulisan video. Sisi terpanjang dibatasi hingga 960 piksel. Langkah ini membantu mengendalikan biaya komputasi dan ukuran keluaran, tetapi detail objek kecil dapat berkurang. Parameter `imgsz=640` mengatur ukuran pemrosesan internal model, bukan janji bahwa video akhirnya berdimensi sama dengan nilai itu.

Kotak hasil mengacu pada frame yang dianalisis setelah penyesuaian ukuran. Jika koordinat hendak dipindahkan ke citra asli, skala perubahan ukuran harus diperhitungkan. Program menggambar anotasi, menulis AVI sementara, lalu memakai `ffmpeg` untuk membuat MP4 H.264 yang sesuai untuk tampilan browser. Audio tidak ikut disalin. Video akhir dibaca kembali untuk memeriksa keterbacaannya; keberhasilan menulis nama file belum cukup membuktikan video dapat diputar.

2.5 Webcam pengguna berada di sisi browser

Runtime Colab adalah komputer yang berbeda dari laptop pengguna. Karena itu, `VideoCapture(0)` pada Python tidak berarti membuka webcam laptop. Jalur opsional menggunakan `getUserMedia` di browser dengan persetujuan pengguna, lalu mengirim gambar JPEG ke Python melalui `output.eval_js`. Dalam contoh ini browser mengambil lima frame terlebih dahulu, baru model menganalisisnya.

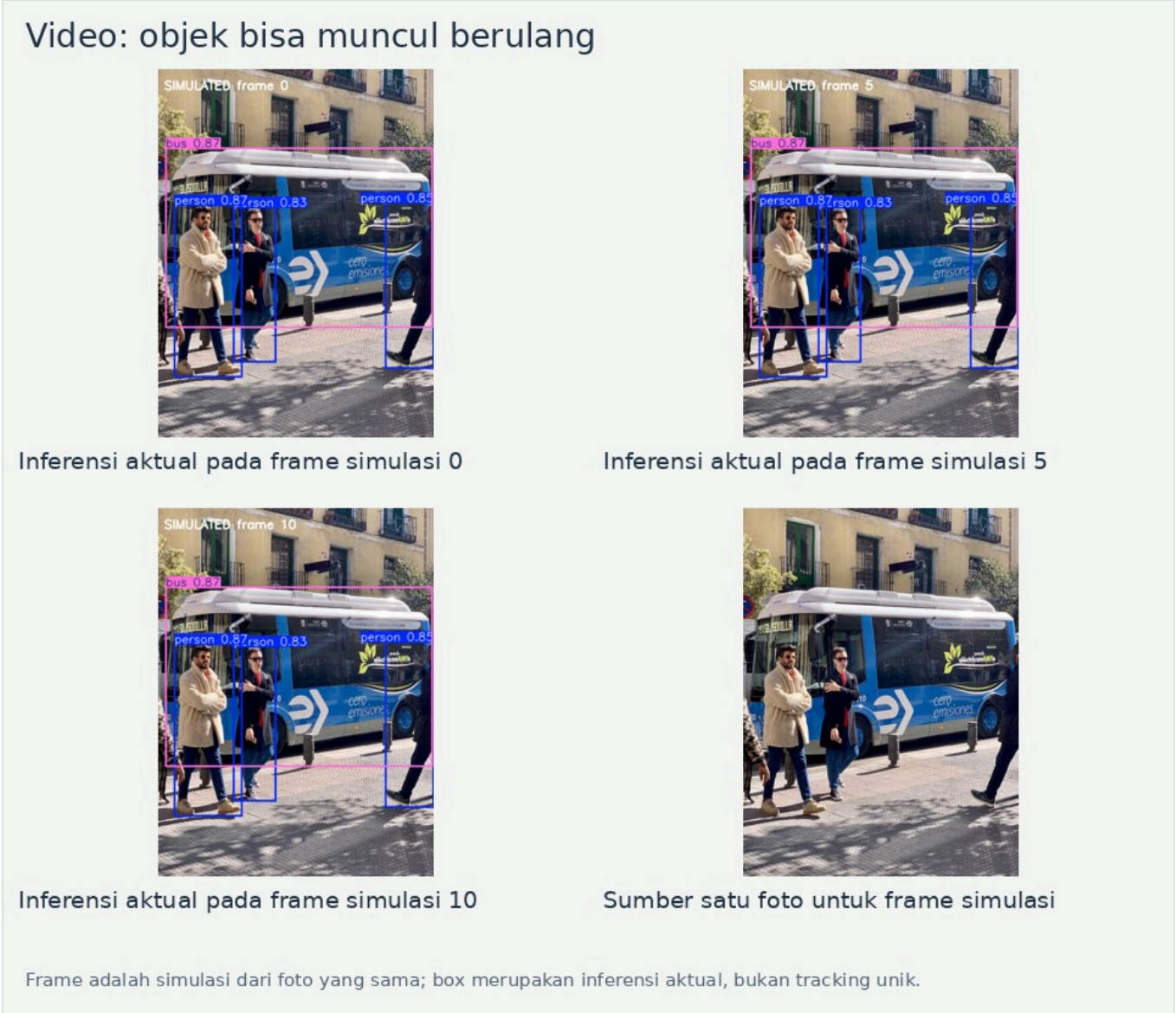
Alur tersebut merupakan capture batch terbatas, bukan aplikasi desktop realtime dan bukan pengukur FPS kamera. Jeda pengambilan, pengiriman gambar, serta waktu model memiliki fungsi yang berbeda. Izin yang ditolak atau perangkat yang sedang digunakan aplikasi lain dapat menggagalkan capture. Blok `finally` menghentikan stream dan membersihkan elemen kamera, termasuk penanganan izin yang terlambat selesai. Penggunaan kamera tetap perlu memperhatikan persetujuan orang yang terlihat dan tidak merekam informasi pribadi tanpa kebutuhan.

2.6 Menilai hasil sebagai satu sistem

Hasil yang masuk akal perlu diperiksa dari masukan sampai keluaran. Bandingkan frame anotasi dengan isi adegan, lalu baca tabel dan laporan waktu menggunakan makna yang tepat. Nol deteksi dapat berarti tidak ada target yang dikenali, bukan otomatis tidak ada target di dunia

nyata. False positive, objek terlewat, dan kotak yang tidak stabil tetap mungkin muncul meskipun video berhasil diputar.

Program juga membandingkan jumlah frame yang terbaca dengan metadata dan membebaskan capture serta writer ketika terjadi kesalahan. Langkah ini membatasi kegagalan yang bisa terdeteksi, bukan menjamin setiap video selalu valid. Codec yang tidak didukung, metadata keliru, keterbatasan runtime, dan perubahan pencahayaan masih membatasi hasil. Dengan membaca batas tersebut, keluaran video menjadi bukti proses yang dapat diperiksa, bukan alasan untuk mengklaim penghitungan unik atau kemampuan realtime yang belum diuji.



Frame adalah simulasi dari foto yang sama; box merupakan inferensi aktual, bukan tracking unik.

Gambar di atas dipertahankan dari materi sumber. Ilustrasi sumber tidak dengan sendirinya membuktikan kinerja pada foto lain; perhatikan keterangan, metode, dan batasannya.

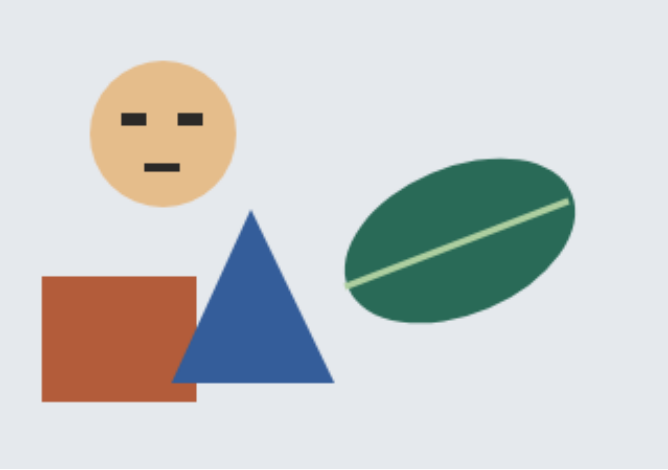
3. Ilustrasi interaktif sebelum praktikum

Simulasi manual: kemunculan temporal bukan objek unik

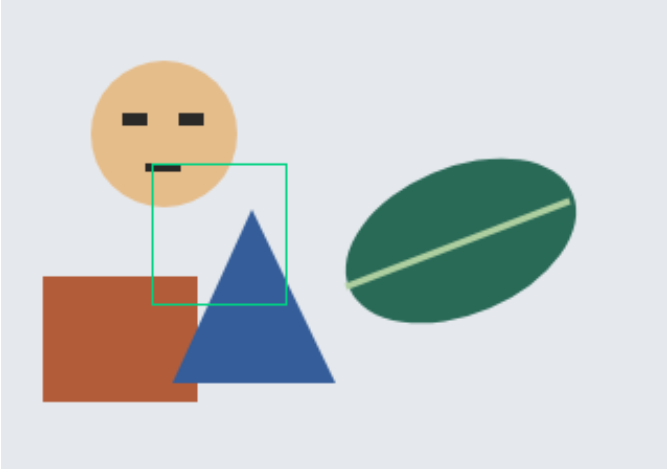
Foto unggahan tetap diam. Satu kotak manual ID-A digeser pada frame simulasi; bukan deteksi YOLO, tracking, atau decode video.

> Pergerakan dan identitas di ilustrasi merupakan contoh buatan. Ini bukan deteksi YOLO, pelacakan nyata, atau decoding video. Tujuannya membedakan kemunculan per frame, total kemunculan, dan identitas konseptual.

Pakai gambar contoh atau pilih foto sendiri dari HP/komputer. Foto dibaca dan diproses **hanya di browser**, tanpa dikirim atau disimpan ke server. Contoh buatan tidak digunakan sebagai bukti hasil eksperimen. Gunakan media berizin dan hindari data sensitif.



Gambar contoh sintetis · citra kerja 320 × 224 piksel



Hasil ilustrasi · baca jenis operasi dan batasannya

Parameter: Indeks frame simulasi (mulai 0) = 30 · FPS sumber yang diasumsikan (bukan pengukuran) = 30 · Penekanan tampilan = frame

Frame 30: 1 kemunculan; kumulatif 31; 1 ID konseptual (ID-A). Waktu simulasi 1.000 s.

Kotak ditentukan manual dan selalu terlihat pada frame 0–30. Waktu = indeks/FPS = 30/30 = 1.000000 s; frame 0 berada di t=0. Kumulatif = 30+1 = 31 kemunculan, bukan jumlah objek dalam foto. ID-A sengaja dipertahankan oleh aturan simulasi, bukan hasil tracking. Jejak abu-abu hanyalah beberapa posisi terdahulu, bukan kotak tambahan pada frame kini. FPS adalah asumsi pengguna, bukan metadata foto, FPS inferensi, atau throughput. Tidak ada deteksi YOLO atau decode video; praktikum video nyata tetap di Colab.

▼ **Angka dan parameter hasil perhitungan ilustrasi**


```
{
  "frameIndex": 30,
  "sourceFpsAssumed": 30,
  "timestampSeconds": 1,
  "perFrameSightings": 1,
  "cumulativeSightings": 31,
  "conceptualUniqueIds": 1,
  "manualId": "ID-A",
  "pixelBox": {
    "x1": 72.47058823529413,
    "y1": 78.39999999999999,
    "x2": 136.47058823529414,
    "y2": 145.6
  },
  "view": "frame",
  "actualDetectionPerformed": false
}
```


Pada PDF/cetakan, ilustrasi ini menjadi cuplikan statis menggunakan gambar contoh. Upload foto dan pengubahan parameter tersedia di versi web; foto pribadi tidak dimasukkan ke PDF unduhan.

4. Video pembelajaran

Object Detection and Tracking with Ultralytics YOLOv8 | Episode 7 — Ultralytics.

Fokus saat menonton: Amati deteksi di video dan asosiasi ID tracking. Bandingkan hitungan deteksi per-frame dengan objek unik.

Buka video di YouTube jika pemutar tidak tersedia

Video referensi berbahasa Inggris. Bila ada setup lokal, API atau versi model berbeda, pelajari konsepnya dan tetap gunakan alur Google Colab pada praktikum bab ini. Ketersediaan pemutar mengikuti YouTube, pemilik video dan jaringan.

▼ **Referensi dan atribusi gambar**

- Ultralytics — Predict, Results dan Boxes

- OpenCV — Video I/O
- Google Colab — FAQ dan keterbatasan runtime

Foto daun: Ανώνυμος Βικιπαιδιστής, Wikimedia Commons, CC BY 3.0; diubah ukuran/intensitas untuk ilustrasi. Foto bus: contoh aset Ultralytics. Foto astronaut: NASA melalui scikit-image. Panel diagram/simulasi ditandai pada caption; bukan hasil eksperimen yang dikarang. Hak penggunaan sumber tetap mengikuti sumber asli.

---

5. Praktikum / latihan Google Colab

Buka Google Colab, buat notebook baru, dan jalankan satu sel kode untuk setiap langkah secara berurutan dengan `Shift` + `Enter`. Variabel dan upload disiapkan di modul ini; tidak mengambil variabel dari bab lain.

Jalankan instalasi dan pemuatan `yolov8n.pt` pada Google Colab, kemudian perhatikan `DEVICE` dan daftar kelas model. Untuk jalur video, upload satu klip berizin melalui sel yang tersedia. Pemanggilan utama `process_video` pada bab ini memakai batas 30–120 detik, sedangkan fungsi dan pemeriksaan upload memungkinkan pengujian awal yang lebih singkat bila parameter minimumnya disesuaikan secara eksplisit.

Sel pemrosesan membaca frame, menyesuaikan ukuran, menjalankan inferensi, menulis anotasi, dan menyusun `frame_table`. Sesudah konversi dan verifikasi video, sel tampilan memperlihatkan laporan, contoh frame, grafik, dan MP4. Baca `source_fps`, `inference_fps`, serta `processing_fps_before_transcode` sebagai ukuran berbeda; kolom jumlah deteksi tetap menghitung kemunculan per frame, bukan identitas objek.

Jalur webcam dapat dijalankan setelah setup tanpa melalui upload dan pemrosesan video. Sel opsional meminta izin browser, mengambil lima frame, lalu menampilkan dan menyimpan anotasi tiap frame. Jika izin tidak tersedia, gunakan jalur video; jangan mengganti jalurnya dengan akses kamera server. Hasil penting perlu disimpan sebelum runtime Colab berakhir karena direktori `/content` bersifat sementara.

Sel 1 — Instalasi Ultralytics

Jalankan sebagai sel tersendiri di Colab; tidak perlu meng-upgrade NumPy. Catat versi paket di sel berikutnya.

```
%pip install -q ultralytics
```

Sel 2 — Model baseline dan direktori keluaran

Bobot yolov8n.pt diunduh saat pemuatan pertama. Ini baseline ringan, bukan pernyataan bahwa model ini yang terbaru. CPU dapat digunakan.

```
from pathlib import Path
import cv2, numpy as np, matplotlib.pyplot as plt
import torch, ultralytics
from ultralytics import YOLO
from google.colab import files
OUTPUT = Path('/content/hasil_yolo')
OUTPUT.mkdir(parents=True, exist_ok=True)
DEVICE = 0 if torch.cuda.is_available() else 'cpu'
model = YOLO('yolov8n.pt')
print('Ultralytics:', ultralytics.__version__, 'Device:', DEVICE)
print('Kelas model:', model.names)
```

Sel 3 — Upload satu file video

Contoh utama memakai file, bukan kamera lokal runtime. Maksimum 150 MB; fungsi menerima klip 1–120 detik agar pengujian awal dapat dilakukan dengan klip singkat.

```
video_files = files.upload()
if len(video_files) != 1:
    raise ValueError('Upload tepat satu file video.')
video_name, video_bytes = next(iter(video_files.items()))
suffix = Path(video_name).suffix.lower()
if suffix not in {'.mp4', '.avi', '.mov', '.mkv', '.webm'}:
    raise ValueError('Gunakan MP4/AVI/MOV/MKV/WebM.')
if not (0 < len(video_bytes) <= 150 * 1024**2):
    raise ValueError('Video kosong atau melebihi 150 MB.')
VIDEO_INPUT = Path('/content') / ('input_video' + suffix)
VIDEO_INPUT.write_bytes(video_bytes)
print('Input:', VIDEO_INPUT)
```

Sel 4 — Pemrosesan frame, hitungan, FPS, dan encoding

Fungsi ini memeriksa capture, metadata, writer dan durasi aktual. Semua frame diproses; keluaran tidak memuat audio. FPS inferensi berbeda dari FPS video sumber.

```
import math, time, shutil, subprocess
import pandas as pd
from collections import Counter

def process_video(source, model, conf, output_dir, device,
classes=None,
                  min_seconds=1.0, max_seconds=120.0):
    if not (0 < conf <= 1):
        raise ValueError('Confidence harus (0,1].')
    if shutil.which('ffmpeg') is None:
        raise RuntimeError('ffmpeg tidak ditemukan pada runtime.')
    output_dir = Path(output_dir); output_dir.mkdir(parents=True,
exist_ok=True)
    cap = cv2.VideoCapture(str(source))
    writer = None
    rows, first_annotated = [], None
    avi = output_dir / 'anotasi_mjpg.avi'
    mp4 = output_dir / 'anotasi.mp4'
    start = time.perf_counter()
    infer_seconds = 0.0
    try:
        if not cap.isOpened():
            raise ValueError('Video gagal dibuka; cek codec atau
konversi input.')
        fps = float(cap.get(cv2.CAP_PROP_FPS))
        width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
        height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
        frame_count = float(cap.get(cv2.CAP_PROP_FRAME_COUNT))
        if not math.isfinite(fps) or not 1 <= fps <= 60:
            raise ValueError('FPS metadata harus valid (1–60);
jangan menebak FPS.')
        if width <= 0 or height <= 0 or width * height > 8_300_000:
            raise ValueError('Dimensi video tidak valid atau
melebihi batas.')
        if not math.isfinite(frame_count) or frame_count <= 0:
            raise ValueError('Jumlah frame metadata tidak valid.')
        estimated_duration = frame_count / fps
        if not min_seconds <= estimated_duration <= max_seconds +
1/fps:
            raise ValueError(f'Durasi metadata
{estimated_duration:.2f}s di luar batas.')
        scale = min(1.0, 960 / max(width, height))
```

```
        out_w = max(2, int(width * scale) // 2 * 2)
        out_h = max(2, int(height * scale) // 2 * 2)
        writer = cv2.VideoWriter(str(avi),
cv2.VideoWriter_fourcc(*'MJPG'),
                                 fps, (out_w, out_h))
        if not writer.isOpened():
            raise RuntimeError('VideoWriter tidak dapat dibuka.')
        index = 0
        limit = math.ceil(max_seconds * fps)
        while True:
            ok, frame = cap.read()
            if not ok:
                break
            if index >= limit:
                raise ValueError('Frame aktual melebihi batas
durasi; input ditolak.')
            frame = cv2.resize(frame, (out_w, out_h))
            if device != 'cpu': torch.cuda.synchronize()
            t0 = time.perf_counter()
            result = model.predict(frame, conf=conf,
classes=classes,
                                   device=device, imgsz=640,
verbose=False)[0]
            if device != 'cpu': torch.cuda.synchronize()
            elapsed = time.perf_counter() - t0
            infer_seconds += elapsed
            counts = Counter(model.names[int(c)] for c in
result.boxes.cls.cpu().tolist())
            annotated = result.plot()
            text = f'frame {index} | detections {len(result.boxes)}'
            cv2.putText(annotated, text, (10, 30),
cv2.FONT_HERSHEY_SIMPLEX,
                        0.6, (0, 255, 255), 2)
            writer.write(annotated)
            if first_annotated is None: first_annotated =
annotated.copy()
            rows.append({'frame': index, 'time_s': index/fps,
                         'detections': len(result.boxes),
'inference_s': elapsed,
                         **{f'count_{name}': counts.get(name, 0)
                            for name in model.names.values()}})
            index += 1
```

```
        actual_duration = index / fps
        if index == 0 or actual_duration + 1/fps < min_seconds:
            raise ValueError('Frame terbaca kosong atau durasi
aktual terlalu pendek.')
        if abs(index - frame_count) > max(2, 0.01 * frame_count):
            raise ValueError('Decode tidak lengkap atau metadata
tidak konsisten; cek file.')
    finally:
        cap.release()
        if writer is not None: writer.release()
    processing_seconds = time.perf_counter() - start
    if not avi.exists() or avi.stat().st_size == 0:
        raise RuntimeError('Video hasil kosong.')
    subprocess.run(['ffmpeg', '-y', '-loglevel', 'error', '-i',
str(avi),
                    '-an', '-c:v', 'libx264', '-pix_fmt', 'yuv420p',
                    '-movflags', '+faststart', str(mp4)],
check=True, timeout=240)
    check = cv2.VideoCapture(str(mp4))
    try:
        valid, _ = check.read()
        output_frames = check.get(cv2.CAP_PROP_FRAME_COUNT)
        if not check.isOpened() or not valid or abs(output_frames -
index) > 2:
            raise RuntimeError('Verifikasi video hasil gagal.')
    finally:
        check.release()
    table = pd.DataFrame(rows)
    table.to_csv(output_dir / 'per_frame.csv', index=False)
    report = {'frames': index, 'source_fps': fps,
'duration_decoded_s': actual_duration,
              'inference_fps': index / infer_seconds,
              'processing_fps_before_transcode': index /
processing_seconds,
              'sum_frame_detections_NOT_unique':
int(table['detections'].sum()),
              'mean_detections_per_frame':
float(table['detections'].mean())}
    return mp4, table, report, first_annotated
```

Sel 5 — Jalankan dan tampilkan laporan video

Sel contoh menggunakan min_seconds=30.0. Untuk latihan awal dengan klip pendek, ubah menjadi 1.0 dan tetap periksa durasi serta metadata yang dilaporkan.

```
from IPython.display import display, Video
mp4_path, frame_table, report, first_frame = process_video(
    VIDEO_INPUT, model, conf=0.35, output_dir=OUTPUT, device=DEVICE,
    min_seconds=30.0, max_seconds=120.0)
display(pd.DataFrame([report]))
display(frame_table.head())
plt.figure(figsize=(10,6)); plt.imshow(cv2.cvtColor(first_frame,
cv2.COLOR_BGR2RGB))
plt.axis('off'); plt.show()
plt.figure(figsize=(12,4)); plt.plot(frame_table['time_s'],
frame_table['detections'])
plt.xlabel('Waktu video (s)'); plt.ylabel('Deteksi per frame');
plt.grid(); plt.show()
display(Video(str(mp4_path), embed=True, width=720))
print('Video:', mp4_path, 'CSV:', OUTPUT / 'per_frame.csv')
print('Jumlah kemunculan deteksi bukan jumlah objek unik.')
```

Sel 6 — Opsional: ambil lima frame kamera browser

Jalankan hanya jika ingin memilih jalur webcam. Izin kamera diperlukan; maksimum lima frame, stream dihentikan dalam finally. Capture berurutan kemudian inferensi: bukan realtime desktop.

```
from google.colab import output
from base64 import b64decode

CAMERA_JS = r"""
async function captureBoundedFrames() {
  let stream = null, expired = false, timer = null;
  const video = document.createElement('video');
  video.setAttribute('playsinline', ''); video.muted = true;
  document.body.appendChild(video);
  const stop = s => s.getTracks().forEach(t => t.stop());
  try {
    const request = navigator.mediaDevices.getUserMedia({video:
{width:640, height:480}, audio:false});
    request.then(s => { if (expired) stop(s); }).catch(() => {});
    const timeout = new Promise((_, reject) => {
```

```
      timer = setTimeout(() => { expired = true; reject(new
Error('Izin kamera timeout')); }, 20000);
    });
    stream = await Promise.race([request, timeout]);
    clearTimeout(timer);
    video.srcObject = stream;
    await Promise.race([video.play(), new Promise((_,reject) =>
      setTimeout(() => reject(new Error('Kamera tidak siap')),
10000))]);
    const canvas = document.createElement('canvas');
    canvas.width = 640; canvas.height = 480;
    const context = canvas.getContext('2d');
    const frames = [];
    for (let i = 0; i < 5; i++) {
      await new Promise(resolve => setTimeout(resolve, 400));
      if (!video.videoWidth) throw new Error('Tidak ada frame
kamera');
      context.drawImage(video, 0, 0, 640, 480);
      frames.push(canvas.toDataURL('image/jpeg', 0.8));
    }
    return frames;
  } finally {
    expired = true; clearTimeout(timer);
    if (stream) stop(stream);
    video.srcObject = null; video.remove();
  }
}
captureBoundedFrames();
"""
def capture_browser_frames():
    encoded_frames = output.eval_js(CAMERA_JS)
    if not isinstance(encoded_frames, list) or not 1 <=
len(encoded_frames) <= 5:
        raise ValueError('Respons kamera tidak valid.')
    frames = []
    for value in encoded_frames:
        if not value.startswith('') or
len(value) > 2_000_000:
            raise ValueError('Data kamera tidak valid.')
        frame =
cv2.imdecode(np.frombuffer(b64decode(value.split(',',1)[1]),
                                           np.uint8),
```

```
cv2.IMREAD_COLOR)
        if frame is None: raise ValueError('Decode kamera gagal.')
        frames.append(frame)
    return frames
webcam_frames = capture_browser_frames()
for i, frame in enumerate(webcam_frames):
    result = model.predict(frame, conf=0.35, device=DEVICE,
verbose=False)[0]
    annotated = result.plot()
    if not cv2.imwrite(str(OUTPUT / f'webcam_{i:02}.jpg'),
annotated):
        raise IOError('Gagal simpan frame kamera.')
    plt.figure(figsize=(9,6)); plt.imshow(cv2.cvtColor(annotated,
cv2.COLOR_BGR2RGB))
    plt.title(f'Frame browser {i}: {len(result.boxes)} deteksi')
    plt.axis('off'); plt.show()
```

Jika terjadi error

Video tidak bisa diputar di browser.

Periksa log ffmpeg, pastikan MP4 H.264 dan yuv420p terbentuk, lalu verifikasi dengan pembacaan ulang.

FPS sumber nol atau NaN.

Input ditolak; ekspor ulang dengan FPS yang jelas. Jangan mengganti metadata diam-diam.

Webcam tidak tersedia.

Periksa izin browser dan kamera; gunakan file video. Jangan memakai VideoCapture(0) di runtime.

Proses terasa lambat.

Gunakan runtime GPU bila tersedia atau uji klip singkat; tetap laporkan batas pengukuran tanpa menjanjikan realtime.

> Runtime Colab dan file upload bersifat sementara. Simpan output penting pada notebook/Drive sebelum sesi berakhir. Jika runtime direset, jalankan ulang sel dari awal. Gunakan media berizin dan hindari unggahan data sensitif tanpa izin.

Pengolahan Citra · Pendahuluan dan daftar seluruh materi