PENGOLAHAN CITRA · MATERI 14

Project Akhir — Sistem Computer Vision berbasis YOLO

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

Sebuah gambar dengan kotak deteksi menunjukkan bahwa model pernah menghasilkan prediksi. Namun, gambar itu belum menjelaskan bagaimana pengguna memasukkan data, apakah file diperiksa, parameter apa yang dipakai, atau di mana hasil disimpan. Sistem computer vision baru terasa utuh ketika semua bagian tersebut terhubung. Pengguna memberikan masukan, program mengolahnya dengan aturan yang jelas, lalu keluaran dapat dibaca dan ditelusuri kembali.

Bayangkan seseorang ingin memeriksa kendaraan pada foto dan klip pendek. Ia perlu memilih jenis input, mengetahui kelas yang tersedia, dan mengatur batas confidence tanpa mengubah algoritme utama. Jika file rusak atau kelas tidak dikenal, program seharusnya berhenti dengan pesan yang sesuai. Menghasilkan tampilan seolah-olah berhasil justru menyembunyikan masalah yang perlu diketahui pengguna.

Bab ini menyatukan pembacaan gambar, pemrosesan video, pemilihan model, anotasi, counting, dan penyimpanan hasil dalam satu alur Google Colab. Sebutan sistem end-to-end berarti jalur dari input sampai output benar-benar tersambung; bukan berarti aplikasi sudah realtime, bebas kesalahan, atau siap diterapkan pada semua kondisi. Batas kemampuan tetap mengikuti data, model, perangkat, dan pengujian yang dilakukan.

Ilustrasi foto interaktif membantu melihat hubungan antara masukan dan keluaran visual sebelum beralih ke video penerapan. Praktikum kemudian memperlihatkan bentuk alur itu melalui form parameter dan fungsi Python yang benar-benar berjalan. Fokus pembahasan bukan membuat daftar fitur sebanyak mungkin, melainkan memahami tanggung jawab setiap tahap, mengenali kegagalan yang mungkin terjadi, serta membaca laporan run secara jujur. Dengan cara ini, model dipahami sebagai satu komponen di dalam sistem, bukan keseluruhan sistem itu sendiri.

2. Pahami konsep secara berurutan

2.1 Kebutuhan pengguna menentukan alur sistem

Sistem dimulai dari kebutuhan yang terbatas dan jelas. Deteksi kendaraan pada foto serta klip pendek membutuhkan keluaran yang berbeda dari penghitungan kendaraan unik sepanjang hari. Kebutuhan pertama dapat menggunakan deteksi per gambar atau frame, sedangkan kebutuhan kedua memerlukan identitas temporal dan aturan counting tambahan. Menjelaskan batas sejak awal mencegah keluaran sederhana diberi arti yang belum didukung program.

Contoh aplikasi menghubungkan pilihan mode, upload, validasi, preprocessing, pemanggilan model, anotasi, tabel, dan penyimpanan. Antarmuka parameter Colab membantu pengguna mengubah pengaturan tanpa membongkar fungsi inti. Jika kontrol form tidak ditampilkan, nilai variabel tetap dapat diedit langsung. Bentuknya adalah notebook interaktif pada runtime Colab, bukan layanan web permanen dengan server dan pengelolaan banyak pengguna.

2.2 Validasi melindungi proses sebelum inferensi

Masukan tidak langsung dikirim ke model. Pada mode gambar, program menerima satu JPG, JPEG, atau PNG, membatasi ukuran berkas dan jumlah piksel, lalu mencoba membaca isinya. Nama file berakhiran gambar belum membuktikan isinya valid. Pada mode video, program juga memeriksa FPS, dimensi, jumlah frame, serta durasi. Berkas yang tidak dapat dibaca dihentikan sebagai kegagalan input, bukan dianggap video kosong yang berhasil diproses.

File disimpan pada lokasi yang dikendalikan aplikasi, sedangkan setiap setup run membuat direktori keluaran tersendiri. Pemisahan ini membantu mencegah hasil lama tertukar dengan proses baru. Validasi yang ada membatasi beberapa risiko dan beban komputasi, tetapi tidak menggantikan pengamanan layanan publik. Media tetap perlu berizin, dan data pribadi sebaiknya tidak diunggah hanya demi mencoba fitur.

2.3 Model aktif menentukan arti kelas dan parameter

`MODEL_SOURCE` memilih baseline `yolov8n.pt` atau bobot custom terpercaya. Model harus bertugas deteksi, dan nama kategori dibaca dari `model.names`. Karena itu, filter kelas mengikuti model yang benar-benar dimuat. Mengetik nama kategori baru pada `TARGET_CLASSES` tidak mengajari model kategori tersebut; nama yang tidak tersedia justru ditolak sebelum inferensi.

`CONFIDENCE` membatasi prediksi yang diteruskan ke keluaran. Ambang lebih tinggi dapat menyingkirkan prediksi lemah sekaligus menghilangkan target yang benar, sehingga tidak otomatis meningkatkan kualitas keseluruhan. Jika memakai model custom, pemilihan bobot dan pengaturan didasarkan pada validation dari data yang dipisahkan berdasarkan scene; held-out test digunakan setelah keputusan dikunci. Memuat file `.pt` juga merupakan batas

kepercayaan: ekstensi dan ukuran tidak membuktikan bobot aman. Contoh ini tidak membenarkan upload model sembarang dari sumber tak dikenal.

2.4 Preprocessing dan anotasi harus memakai ruang yang sama

Gambar diperkecil secara proporsional bila sisi terpanjangnya melebihi 960 piksel. Model kemudian memproses citra tersebut dengan ukuran internal yang ditentukan `imgsz`. Resize dapat mengurangi biaya komputasi, tetapi juga mengurangi detail target kecil. Tidak ada enhancement tambahan yang dianggap selalu memperbaiki hasil. Perubahan masukan perlu dipahami karena memengaruhi isi yang dilihat model.

Pada mode gambar, tabel menyimpan kelas, confidence, serta `x1`, `y1`, `x2`, dan `y2` dalam piksel citra setelah resize. Ini berbeda dari label training YOLO yang memakai pusat dan ukuran ternormalisasi. Menggambar koordinat hasil pada gambar asli tanpa menyesuaikan skala dapat menghasilkan kotak yang salah posisi. OpenCV mempertahankan urutan BGR dalam pemrosesan, sedangkan tampilan Matplotlib memakai konversi RGB agar warna tidak tertukar.

2.5 Tiga mode input mempunyai batas keluaran berbeda

Mode gambar menghasilkan anotasi, tabel bbox, dan hitungan kelas pada satu citra. Tabel dengan nol baris tetap mempunyai nama kolom sehingga tidak rancu dengan proses yang gagal membentuk keluaran. Nol deteksi hanya menyatakan tidak ada kotak yang lolos keputusan model pada pengaturan tersebut. Target kecil, tertutup, atau di luar kategori model mungkin masih ada.

Mode video menjalankan proses serupa berulang pada frame, kemudian membuat tabel temporal dan MP4 tanpa audio. FPS sumber menyatakan laju pemutaran, FPS inferensi menyatakan laju kerja model, sedangkan throughput pipeline sebelum transcoding mencakup pekerjaan pendukung. Ketiganya tidak boleh dipertukarkan. Penjumlahan bbox lintas frame adalah kemunculan berulang, bukan jumlah objek unik. Tracking dan aturan lintasan dibutuhkan bila keluaran yang diinginkan berkaitan dengan identitas atau perlintasan garis.

Mode `webcam_browser` mengambil lima frame lewat izin kamera pada browser, lalu menjalankan inferensi sesudah capture. Python Colab tidak membuka webcam laptop melalui `VideoCapture(0)`. Alur batch ini bukan realtime desktop. Stream dihentikan melalui cleanup, dan gangguan izin atau perangkat dapat membuat capture gagal. Dengan demikian, webcam adalah jalur input tambahan yang memiliki batas jelas, bukan bukti sistem sudah memenuhi kebutuhan pemantauan langsung.

2.6 Pengujian dan laporan membuat hasil dapat dipercaya

Pengujian tidak berhenti pada satu gambar yang mudah. Input valid, file tidak sesuai, citra tanpa target, dan objek sulit memperlihatkan perilaku yang berbeda. Periksa apakah pesan

kegagalan sesuai tahapnya serta apakah hasil yang disimpan dapat dibaca. Pada video, capture dan writer dibebaskan ketika proses berakhir atau mengalami exception. MP4 dibaca kembali setelah konversi, sehingga keberadaan nama file saja tidak diperlakukan sebagai bukti keberhasilan.

Sel akhir menyimpan `config.json` dan `report.csv` untuk menghubungkan pengaturan dengan hasil run. Daftar file beserta ukurannya membantu menemukan keluaran kosong, tetapi ukuran yang tidak nol tidak menjamin prediksi benar. Kebenaran tetap memerlukan pemeriksaan terhadap objek nyata atau anotasi yang dapat dipercaya. Runtime sementara, keterbatasan GPU, jaringan, serta pergeseran kondisi gambar masih membatasi sistem. Aplikasi end-to-end yang baik menjelaskan apa yang dijalankan dan apa yang belum diuji, tanpa mengarang hasil atau menyamakan demo berhasil dengan kesiapan operasional.



Contoh komponen project: inference YOLOv8n benar-benar dijalankan; UI ada di praktikum.

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

---

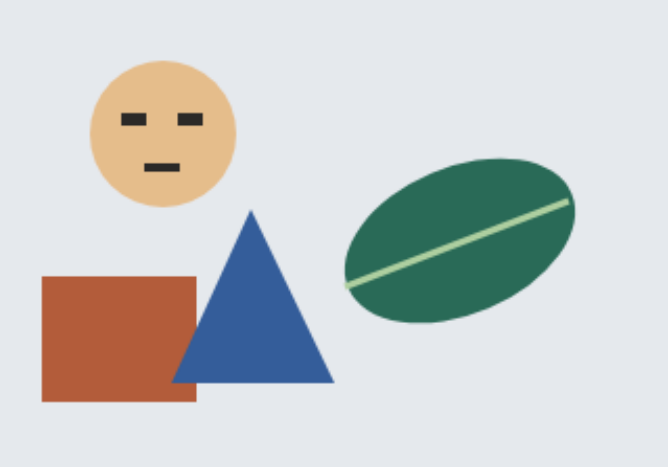
3. Ilustrasi interaktif sebelum praktikum

Alur reproduksibel: preprocessing nyata dan jejak konfigurasi

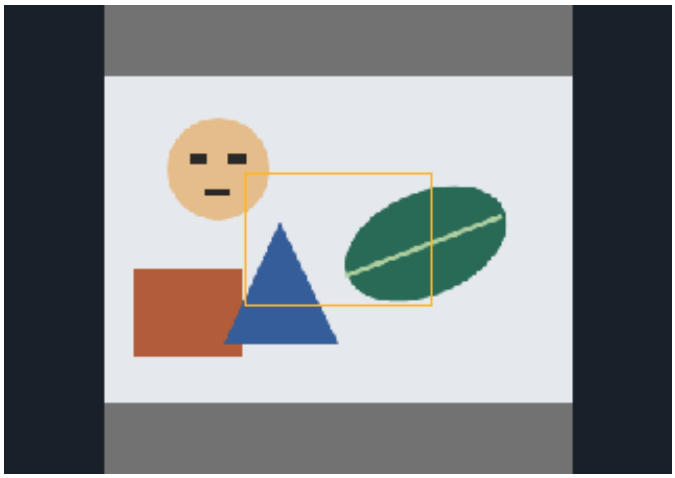
Resize + letterbox benar-benar dihitung dari piksel unggahan. Kotak hanya ROI manual untuk audit koordinat; confidence hanya dicatat, tanpa model browser atau hasil deteksi.

Ilustrasi memperlihatkan tahap dan parameter pengolahan sebagai kontrak sistem. Ini tidak menjalankan model YOLO di browser. Inferensi gambar/video dan laporan run model yang sebenarnya tersedia pada praktikum Colab.

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: Sisi target letterbox (px) = 320 · Pusat X ROI manual (%) = 50 · Pusat Y ROI manual (%) = 50 · Confidence rencana Colab (hanya konfigurasi) = 0.35

320 × 224 → 320 × 224 + padding → 320 × 320. ROI manual: 11520 piksel; rerata keabuan 169.27 / 255.

Alur aktual: validasi RGBA → resize nearest-neighbor (pusat piksel) → letterbox RGB 114 → pemetaan ROI → pratinjau. Skala ideal 1.000000; setelah pembulatan dimensi: sx=1.000000, sy=1.000000. Padding kiri/atas/kanan/bawah = 0/48/0/48 px. ROI sumber [x1,y1,x2,y2] = [96, 67, 224, 157], ROI target = [96.000, 115.000, 224.000, 205.000]; x′=sx·x+padLeft, y′=sy·y+padTop. Rerata ROI dihitung dari RGB sumber dengan (77R+150G+29B)/256, tanpa pembobotan alpha. Kotak jingga hanya ROI manual, bukan prediksi bbox. Target persegi ditampilkan proporsional di bidang 320 × 224; bidang gelap luar bukan padding model. Confidence 0.35 hanya rekaman parameter rencana, tidak menyaring piksel atau ROI. Tidak ada model browser, training, inferensi, hitungan objek, skor prediksi, atau pengukuran FPS. Training/inferensi nyata dan konfigurasi model lengkap tetap di Colab; preprocessing pedagogis ini bukan jaminan identik dengan pipeline YOLO. Konfigurasi untuk diulang bersama gambar yang sama tersedia pada panel angka dan parameter di bawah ini.

▼ **Angka dan parameter hasil perhitungan ilustrasi**


```
{
  "imageWidth": 320,
  "imageHeight": 224,
  "imgsz": 320,
  "scale": 1,
  "scaleX": 1,
  "scaleY": 1,
  "resizedWidth": 320,
  "resizedHeight": 224,
  "padLeft": 0,
  "padTop": 48,
  "padRight": 0,
  "padBottom": 48,
  "sourceROI": {
    "x1": 96,
    "y1": 67,
    "x2": 224,
    "y2": 157
  },
  "targetROI": {
    "x1": 96,
    "y1": 115,
    "x2": 224,
    "y2": 205
  },
  "previewROI": {
    "x1": 115.2,
    "y1": 80.5,
    "x2": 204.8,
    "y2": 143.5
  },
  "roiPixelCount": 11520,
  "roiMeanGray": 169.26918131510416,
  "config": {
    "engine": "citra-preprocess-v1",
    "inputWidth": 320,
    "inputHeight": 224,
    "imgsz": 320,
    "confidence": 0.35,
```

```
    "confidenceApplied": false,
    "interpolation": "nearest-neighbor pixel-center",
    "paddingRGB": [
      114,
      114,
      114
    ],
    "sourceROI": {
      "x1": 96,
      "y1": 67,
      "x2": 224,
      "y2": 157
    }
  },
  "actualInferencePerformed": 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

How to Perform Real-Time Object Counting with Ultralytics YOLO11 | Apples on a Moving Conveyor Belt — Ultralytics.

Fokus saat menonton: Amati penerapan penghitung objek sebagai contoh project. Fokus pada alur input–deteksi–tracking–keluaran dan evaluasi; video menggunakan YOLO11 sedangkan contoh kuliah memakai YOLOv8.

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, lalu atur `INPUT_MODE`, `MODEL_SOURCE`, `CONFIDENCE`, dan `TARGET_CLASSES` pada sel parameter. Baseline dimuat langsung; pilihan custom meminta bobot deteksi dari sumber yang dipercaya. Perhatikan kelas model, perangkat, dan direktori `OUTPUT` sebelum menjalankan sel helper. Semua inferensi berlangsung nyata di runtime Colab, bukan pada ilustrasi browser halaman materi.

Sel helper mendefinisikan pembacaan gambar, deteksi, pemrosesan video, dan capture kamera. Sel aplikasi baru memanggil jalur yang dipilih: upload gambar menghasilkan anotasi dan tabel bbox; upload video menghasilkan MP4, tabel per frame, grafik, dan laporan; webcam mengambil batch frame browser. Ketika mengganti mode atau parameter, jalankan kembali setup dan sel-sel berikutnya secara berurutan agar fungsi memakai konfigurasi yang baru.

Sel ringkasan membentuk `report_table`, menyimpan konfigurasi serta laporan, lalu memeriksa adanya file kosong. Cocokkan keluaran dengan input run yang sedang diproses, terutama ketika membandingkan gambar dan video. Direktori run yang berbeda membantu penelusuran, tetapi tetap berada pada runtime sementara; simpan hasil penting sebelum sesi berakhir. Error yang muncul perlu dibaca sebagai informasi tentang tahap yang gagal, bukan diganti dengan gambar hasil dari proses lain.

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 — Antarmuka parameter Colab dan model

Parameter #@param menjadi form di Colab (atau edit nilainya langsung). Pilih mode dan confidence, jalankan setup lalu sel berikutnya. Custom hanya bobot .pt dari training sendiri yang Anda percayai; pemuatan bobot tak tepercaya berisiko, bukan input publik sembarang.

```
#@title Parameter aplikasi
INPUT_MODE = 'gambar' #@param ['gambar', 'video', 'webcam_browser']
CONFIDENCE = 0.35 #@param {type:'number'}
MODEL_SOURCE = 'baseline' #@param ['baseline', 'custom_terpercaya']
TARGET_CLASSES = '' #@param {type:'string'}

from pathlib import Path
from collections import Counter
import io, time, uuid, json
import cv2, numpy as np, matplotlib.pyplot as plt, pandas as pd
from PIL import Image
import torch, ultralytics
from ultralytics import YOLO
from google.colab import files, output
from IPython.display import display, Video
if INPUT_MODE not in {'gambar','video','webcam_browser'}:
    raise ValueError('Mode input tidak valid.')
if not 0 < CONFIDENCE <= 1:
    raise ValueError('Confidence harus di rentang (0,1].')
DEVICE = 0 if torch.cuda.is_available() else 'cpu'
OUTPUT = Path('/content/hasil_project') / uuid.uuid4().hex[:8]
OUTPUT.mkdir(parents=True, exist_ok=False)
if MODEL_SOURCE == 'baseline':
    MODEL_LABEL = 'yolov8n.pt'
elif MODEL_SOURCE == 'custom_terpercaya':
    trusted = files.upload()
    if len(trusted) != 1: raise ValueError('Upload satu best.pt
milik kelompok.')
    weight_name, weight_bytes = next(iter(trusted.items()))
    if Path(weight_name).suffix.lower() != '.pt' or not 0 <
len(weight_bytes) <= 100*1024**2:
        raise ValueError('Bobot harus .pt, maksimum 100 MB.')
    MODEL_LABEL = str(OUTPUT/'custom_best.pt')
    Path(MODEL_LABEL).write_bytes(weight_bytes)
else:
    raise ValueError('Sumber model tidak valid.')
```

```
model = YOLO(MODEL_LABEL)
if model.task != 'detect': raise ValueError('Gunakan model object
detection, bukan task lain.')
requested = [name.strip() for name in TARGET_CLASSES.split(',') if
name.strip()]
name_to_id = {name:int(idx) for idx,name in model.names.items()}
if set(requested)-set(name_to_id):
    raise ValueError(f'Kelas tidak tersedia: {set(requested)-
set(name_to_id)}')
CLASS_FILTER = [name_to_id[name] for name in requested] if requested
else None
print('Model:', MODEL_LABEL, 'device:', DEVICE, 'Ultralytics:',
ultralytics.__version__)
print('Kelas model:',model.names,'filter:', requested or 'semua')
print('Output:', OUTPUT)
```

Sel 3 — Validasi, preprocessing dan deteksi gambar

Helper memeriksa ukuran sebelum decode, resize proporsional maksimum 960 piksel, inferensi, counting, menyimpan dan menampilkan. Koordinat tabel merujuk gambar setelah resize.

```
def decode_image(name, content):
    if Path(name).suffix.lower() not in {'.jpg','.jpeg','.png'}:
        raise ValueError('Gambar wajib JPG/JPEG/PNG.')
    if not 0 < len(content) <= 20*1024**2:
        raise ValueError('Gambar kosong atau melebihi 20 MB.')
    with Image.open(io.BytesIO(content)) as header:
        if header.width*header.height > 20_000_000:
            raise ValueError('Gambar melebihi 20 megapiksel.')
        header.verify()
    image =
cv2.imdecode(np.frombuffer(content,np.uint8),cv2.IMREAD_COLOR)
    if image is None: raise ValueError('Decode gambar gagal.')
    return image

def detect_and_show(image, tag):
    h,w = image.shape[:2]
    scale = min(1.0,960/max(h,w))
    if scale < 1:
        image = cv2.resize(image,
```

```
(max(1,round(w*scale)),max(1,round(h*scale))))
    if DEVICE != 'cpu': torch.cuda.synchronize()
    t0 = time.perf_counter()
    result = model.predict(image, conf=CONFIDENCE,
classes=CLASS_FILTER,
                           imgsz=640, device=DEVICE, verbose=False)
[0]
    if DEVICE != 'cpu': torch.cuda.synchronize()
    elapsed = time.perf_counter()-t0
    annotated = result.plot()
    destination = OUTPUT/f'{tag}_annotated.jpg'
    if not cv2.imwrite(str(destination),annotated): raise
IOError('Simpan gambar gagal.')
    rows=[]; counts=Counter()
    for box in result.boxes:
        idx=int(box.cls.item()); name=model.names[idx];
counts[name]+=1
        x1,y1,x2,y2=box.xyxy[0].cpu().tolist()

rows.append({'class':name,'confidence':float(box.conf.item()),
                     'x1':x1,'y1':y1,'x2':x2,'y2':y2})
    table=pd.DataFrame(rows,columns=
['class','confidence','x1','y1','x2','y2'])
    table.to_csv(OUTPUT/f'{tag}_bbox.csv',index=False)
    plt.figure(figsize=(10,6));
plt.imshow(cv2.cvtColor(annotated,cv2.COLOR_BGR2RGB))
    plt.title(f'{tag}: {len(result.boxes)} deteksi');
plt.axis('off'); plt.show()
    display(table)
    print('Hitungan per kelas:',dict(counts))
    return
{'input':tag,'detections':len(result.boxes),'inference_seconds':elap
sed,

'output_image':str(destination),'coordinate_space':'resized image
pixels'}
```

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 — Helper webcam browser opsional

Hanya definisi fungsi pada sel ini. Sel aplikasi memanggilnya bila mode webcam_browser dipilih. Lima frame, izin browser, timeout dan finally stop track; tidak menggunakan VideoCapture(0).

```
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
```

Sel 6 — Jalankan aplikasi end-to-end

Input upload muncul sesuai mode. Gambar dan video ditampilkan otomatis; webcam browser adalah batch terbatas. Kesalahan dihentikan dengan pesan, bukan hasil yang direkayasa.

```
reports = []
if INPUT_MODE == 'gambar':
    inputs = files.upload()
    if len(inputs) != 1: raise ValueError('Upload tepat satu
gambar.')
    input_name, content = next(iter(inputs.items()))
    image = decode_image(input_name,content)
    reports.append(detect_and_show(image,'gambar'))
elif INPUT_MODE == 'video':
    inputs = files.upload()
    if len(inputs) != 1: raise ValueError('Upload tepat satu
```

```
video.')
    input_name, content = next(iter(inputs.items()))
    suffix = Path(input_name).suffix.lower()
    if suffix not in {'.mp4','.mov','.avi','.mkv','.webm'}:
        raise ValueError('Format video tidak didukung.')
    if not 0 < len(content) <= 150*1024**2:
        raise ValueError('Video kosong atau melebihi 150 MB.')
    source = OUTPUT/('input'+suffix); source.write_bytes(content)
    video_path, frame_table, report, first_frame = process_video(

source,model,conf=CONFIDENCE,output_dir=OUTPUT,device=DEVICE,
        classes=CLASS_FILTER,min_seconds=1.0,max_seconds=120.0)
    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 (s)'); plt.ylabel('Deteksi per frame');
plt.grid(); plt.show()
    display(Video(str(video_path),embed=True,width=720))

reports.append({'input':'video','output_video':str(video_path),**rep
ort})
else:
    browser_frames = capture_browser_frames()
    for index, frame in enumerate(browser_frames):
        reports.append(detect_and_show(frame,f'webcam_{index:02}'))
    print('Capture batch selesai; stream kamera dihentikan. Ini
bukan realtime desktop.')
```

Sel 7 — Ringkasan, konfigurasi dan verifikasi output

Ringkasan ini berasal dari proses nyata. Run mode gambar dan video untuk demonstrasi; saat mengganti parameter ulangi setup serta sel aplikasi dan simpan kedua output run pada notebook.

```
report_table = pd.DataFrame(reports)
if report_table.empty: raise RuntimeError('Aplikasi tidak
menghasilkan laporan.')
display(report_table)
```

```
config =
{'mode':INPUT_MODE,'model':MODEL_LABEL,'confidence':CONFIDENCE,
          'classes':requested or 'semua','device':str(DEVICE),
          'ultralytics_version':ultralytics.__version__,
          'preprocessing':'resize proporsional sisi maksimum 960,
tanpa enhancement',
          'counting_note':'jumlah bbox per gambar/frame, bukan objek
unik antarframe'}
(OUTPUT/'config.json').write_text(json.dumps(config,indent=2,ensure_
ascii=False),encoding='utf-8')
report_table.to_csv(OUTPUT/'report.csv',index=False)
output_files=[p for p in OUTPUT.iterdir() if p.is_file()]
if not output_files or any(p.stat().st_size == 0 for p in
output_files):
    raise RuntimeError('Terdapat output kosong; periksa proses.')
display(pd.DataFrame([{'file':p.name,'bytes':p.stat().st_size} for p
in output_files]))
print('Run tersimpan:',OUTPUT)
print('Periksa keluaran aktual dan ringkasan konfigurasi pada
notebook.')
```

Jika terjadi error

Model custom tidak dimuat atau task bukan detect.

Pakai best.pt dari training deteksi sendiri yang dipercaya; periksa file dan task. Jangan memuat bobot tak dikenal.

Mode berubah tetapi input lama tetap dipakai.

Ulangi setup parameter dan sel aplikasi; setiap run mempunyai direktori unik.

Video belum memenuhi batas atau codec tidak terbaca.

Ekspor ulang klip 1–120 detik dengan FPS 1–60 dan format yang terbaca; aplikasi tidak menebak metadata.

Notebook tampak selesai tetapi output kosong.

Periksa exception dan direktori run. Pastikan hasil benar-benar dihasilkan dari proses, bukan mengganti keluaran yang gagal dengan hasil buatan.

> 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.
