PENGOLAHAN CITRA · MATERI 13

Custom Dataset dan Training 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

Model deteksi siap pakai hanya mengenali kategori yang dipelajarinya. Ketika kebutuhan kita berbeda, misalnya membedakan beberapa jenis benda khusus di ruang kerja, mengganti tulisan nama kelas pada gambar tidak cukup. Model memerlukan contoh yang relevan beserta penjelasan tentang benda mana yang harus ditemukan. Kumpulan gambar dan penjelasan itulah yang menjadi dasar dataset deteksi khusus.

Bayangkan dua orang diminta menandai benda yang sama. Orang pertama menggambar kotak hanya pada benda yang terlihat utuh, sedangkan orang kedua juga menandai benda yang terpotong. Tanpa aturan bersama, dataset membawa petunjuk yang saling bertentangan. Model kemudian belajar dari ketidakkonsistenan tersebut. Karena itu, pekerjaan penting sebelum training adalah mendefinisikan kelas, memastikan izin sumber gambar, dan membuat anotasi yang dapat ditinjau kembali.

Masalah lain muncul ketika foto yang sangat mirip tersebar di data belajar dan data pengujian. Hasil evaluasi bisa tampak meyakinkan karena model bertemu suasana yang hampir sama, bukan karena sudah memahami variasi baru. Bab ini membedakan fungsi train, validation, dan test, lalu menjelaskan mengapa gambar dari scene atau sesi pengambilan yang sama sebaiknya tetap berada dalam satu bagian.

Alurnya bergerak dari contoh gambar menuju label, pemeriksaan dataset, pembagian scene, training, dan pembacaan hasil. Ilustrasi membantu memahami posisi kotak, sedangkan video memperlihatkan gambaran proses pelatihan. Praktikum berikutnya benar-benar menjalankan training di Google Colab menggunakan dataset yang diunggah. Angka precision, recall, dan mAP baru mempunyai arti sebagai hasil eksperimen setelah proses itu berjalan; halaman materi tidak menggantinya dengan keluaran latihan yang dibuat-buat.

2. Pahami konsep secara berurutan

2.1 Mulai dari definisi kelas dan keragaman data

Kelas perlu mempunyai arti yang dapat diterapkan secara konsisten pada gambar. Nama yang terdengar jelas belum tentu cukup: aturan tentang objek tertutup, terpotong, atau terlalu kecil juga perlu dipahami. Dengan kebijakan yang sama, dua peninjau lebih mungkin menghasilkan anotasi yang sejalan. Kategori sebaiknya dapat dibedakan dari informasi visual yang memang tersedia, bukan dari ciri yang tidak terlihat di foto.

Contoh kode menerima 100–200 gambar dengan satu hingga tiga kelas. Batas ini menjaga cakupan latihan tetap terkelola, bukan menjamin kecukupan data untuk semua penerapan. Banyak foto dari satu sudut tidak menggantikan variasi latar, jarak, cahaya, dan scene. Sumber serta izin dicatat dalam `LICENSE.md` dan `manifest.csv`. Pemeriksaan bahwa kolom izin terisi tidak membuktikan izin tersebut sah; peninjauan sumber dan perlindungan identitas pribadi tetap diperlukan.

2.2 Anotasi menyatakan apa dan di mana

Label deteksi YOLO menyimpan satu baris untuk setiap objek target: `class_id x_center y_center width height`. ID kelas dimulai dari nol dan mengikuti urutan `classes.json`. Mengubah urutan nama tanpa memperbaiki ID akan mengubah makna label. Empat nilai berikutnya menyatakan pusat dan ukuran kotak, bukan dua sudut kotak seperti keluaran `xyxy` saat inferensi.

Koordinat dinormalisasi terhadap dimensi gambar. Pusat horizontal dan lebar dibagi lebar gambar; pusat vertikal dan tinggi dibagi tinggi gambar. Dengan begitu, nilai menggambarkan posisi relatif. Ukuran harus positif dan seluruh batas kotak tetap berada di dalam gambar. Gambar dan label mempunyai stem sama agar pasangannya jelas. Anotasi yang sah secara angka masih bisa salah secara makna, misalnya kelas tertukar, kotak terlalu longgar, atau target yang tidak ditandai.

2.3 Negatif yang sah dan pemeriksaan sebelum belajar

Gambar negatif adalah gambar yang sudah diperiksa dan memang tidak mengandung target menurut definisi kelas. Ia bukan gambar yang belum sempat dilabeli. Dalam kontrak contoh, label kosong hanya diterima jika manifest menyatakan `negative=true` dan `reviewed=true`. Untuk gambar positif, seluruh target yang memenuhi kebijakan perlu diberi kotak agar model tidak menerima petunjuk yang membingungkan.

ZIP diperiksa sebelum dataset digunakan. Program membatasi jalur ekstraksi, ukuran, jenis berkas, pasangan gambar-label, token kelas, serta nilai koordinat. Pemeriksaan ini membantu menolak arsip berbahaya atau struktur yang tidak konsisten. Duplikat piksel identik juga ditolak,

tetapi foto yang nyaris sama belum tentu terdeteksi. Audit visual menampilkan beberapa ground truth yang diunggah; kotak pada tahap ini adalah anotasi manusia, bukan bukti model sudah berhasil memprediksi.

2.4 Train, validation, dan test mempunyai peran berbeda

Train dipakai untuk memperbarui bobot model. **Validation** memberi umpan balik untuk memilih bobot dan keputusan eksperimen. **Test** ditahan sebagai pemeriksaan akhir setelah keputusan tersebut dikunci. Ketiganya bukan nama folder yang boleh dipertukarkan. Bila test berulang kali dipakai memilih konfigurasi, informasi pengujian ikut memengaruhi pengembangan dan hasil akhirnya tidak lagi independen.

Pembagian perlu memperhatikan hubungan antargambar. Frame berdekatan dari satu video sering memperlihatkan objek dan latar hampir sama. Memasukkannya ke bagian berbeda membuat kebocoran informasi, atau *leakage*. Karena itu, `scene_id` mengelompokkan gambar yang saling berkaitan agar seluruh kelompok masuk satu split. Pengelompokan perlu mengikuti kondisi pengambilan yang nyata; mengganti nama scene tidak membuat dua foto yang hampir identik menjadi independen.

Fungsi `split_by_scene` mencoba pembagian yang mendekati proporsi 70/15/15 sambil menjaga kehadiran semua kelas. Ukuran kelompok berbeda sehingga jumlah gambar aktual perlu dibaca dari laporan. Kode memerlukan sedikitnya enam scene dan menyarankan lebih banyak. Jika pencarian gagal menemukan pembagian yang sesuai, penyelesaiannya adalah menambah scene independen yang relevan, bukan memecah adegan serupa untuk mempercantik angka.

2.5 Fine-tuning mengubah bobot, bukan sekadar label tampilan

Setelah validasi dan split, program menyusun direktori serta `data.yaml` yang menghubungkan lokasi gambar dan nama kelas. Training memulai dari `yolov8n.pt`, kemudian memperbarui bobot menggunakan dataset khusus. Ini berbeda dari inferensi: pada inferensi, model hanya menghasilkan prediksi tanpa belajar dari gambar yang sedang ditampilkan. Kualitas contoh dan anotasi memengaruhi apa yang dapat dipelajari.

Konfigurasi awal menggunakan 30 epoch, `imgsz=640`, dan batch empat, dengan penghentian dini yang dapat membuat proses berhenti sebelum batas epoch. Angka tersebut adalah pengaturan percobaan, bukan resep optimal atau janji kualitas. GPU membantu waktu komputasi, tetapi ketersediaannya mengikuti runtime Colab. Program memeriksa keberadaan `best.pt` hasil training; bobot terbaik dipilih berdasarkan validation, bukan dari skor test. Versi perangkat lunak, perangkat komputasi, seed, serta pengaturan tetap penting untuk menelusuri run.

2.6 Membaca metrik bersama contoh kesalahan

Precision menjelaskan bagian prediksi positif yang benar, sedangkan recall menjelaskan bagian target beranotasi yang berhasil ditemukan. Penilaian juga membutuhkan kesesuaian kelas dan lokasi kotak. `mAP50` merangkum average precision pada ambang IoU 0,50; `mAP50_95` merangkumnya pada beberapa ambang IoU sehingga lebih menuntut ketepatan posisi. Nilai agregat tidak menggantikan pemeriksaan jumlah contoh dan hasil tiap kelas.

Kurva loss membantu memahami proses belajar. Loss train yang membaik sementara validation memburuk dapat mengindikasikan overfitting, tetapi masalah label dan perbedaan distribusi juga perlu diperiksa. Sel evaluasi menjalankan validation dan test secara terpisah, kemudian menampilkan riwayat serta beberapa prediksi test. Prediksi itu tetap harus dibandingkan dengan ground truth. Hasil nyata dapat rendah atau tidak stabil pada dataset kecil; tanpa run yang selesai, tidak ada dasar untuk menyebut model akurat atau siap digunakan pada kondisi baru.



Diagram alur; bukan klaim sebuah dataset atau training sudah tersedia.

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

---

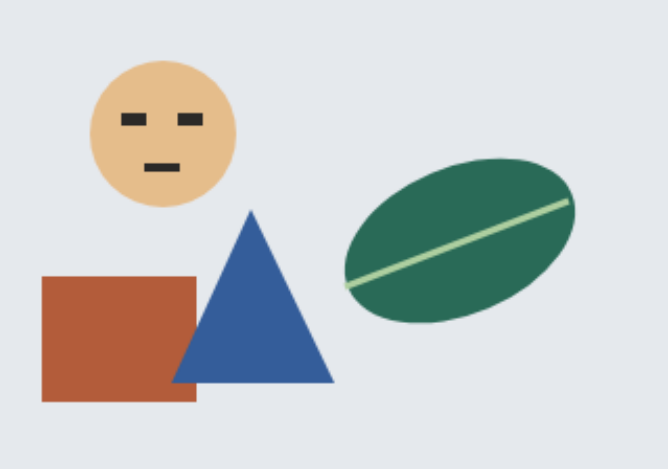
3. Ilustrasi interaktif sebelum praktikum

Pembantu anotasi manual: koordinat YOLO ternormalisasi

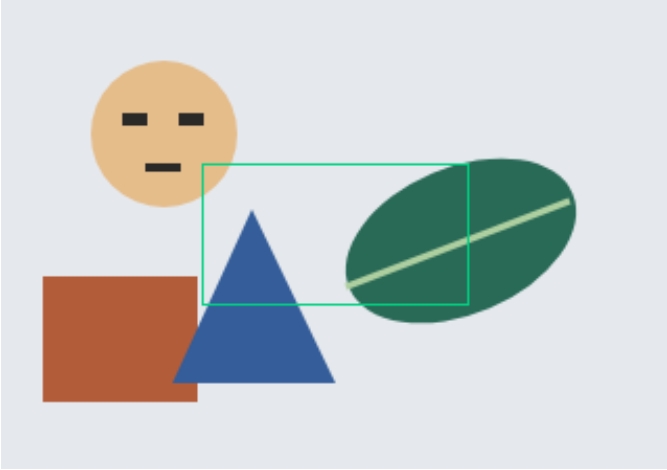
Atur satu kotak secara manual di atas gambar unggahan. Konversi koordinat ini nyata, tetapi bukan training atau prediksi model; ID kelas contoh tetap 0.

> Kotak diatur secara manual untuk mempelajari koordinat anotasi YOLO. Ilustrasi ini tidak melatih model dan tidak menjamin anotasi sudah benar. Tetap tinjau kesesuaian kotak dan kelas dengan objek pada foto.

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: Pusat X yang diminta (%) = 50 · Pusat Y yang diminta (%) = 50 · Lebar kotak (%) = 40 · Tinggi kotak (%) = 30

Label YOLO manual: 0 0.500000 0.500000 0.400000 0.300000

Gambar 320 × 224 px. Sudut kotak kontinu (x1,y1,x2,y2) = (96.000, 78.400, 224.000, 145.600) px; tepi kanan/bawah boleh sama dengan lebar/tinggi citra. xc=(x1+x2)/(2W), yc=(y1+y2)/(2H), w=(x2−x1)/W, h=(y2−y1)/H. Nilai efektif: 0.500000, 0.500000, 0.400000, 0.300000. Tidak perlu penyesuaian pusat. Baris anotasi (kelas 0): 0 0.500000 0.500000 0.400000 0.300000. Kotak adalah pilihan pengguna, bukan ground truth yang otomatis benar: periksa target, definisi kelas, seluruh objek, dan izin gambar. Ini pembantu anotasi, bukan training atau inferensi YOLO; training dan audit dataset tetap di Colab.

▼ **Angka dan parameter hasil perhitungan ilustrasi**


```
{
  "imageWidth": 320,
  "imageHeight": 224,
  "classId": 0,
  "normalized": {
    "xc": 0.5,
    "yc": 0.5,
    "width": 0.4,
    "height": 0.3
  },
  "pixelBox": {
    "x1": 96,
    "y1": 78.39999999999999,
    "x2": 224,
    "y2": 145.6
  },
  "annotationLine": "0 0.500000 0.500000 0.400000 0.300000",
  "centerAdjusted": false,
  "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

How to Train Ultralytics YOLOv8 models on Your Custom Dataset in Google Colab | Episode 3 — Ultralytics.

Fokus saat menonton: Ikuti anotasi dataset, konfigurasi dan training di Google Colab; bandingkan praktik split dengan pencegahan leakage pada bab ini.

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 — Format dataset deteksi YOLO
- Ultralytics — Train
- Ultralytics — Validation metrics
- 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.

Di Colab, jalankan instalasi lalu upload ZIP dengan folder `images` dan `labels`, serta `classes.json`, `manifest.csv`, dan `LICENSE.md` pada tingkat teratas arsip. Sel kontrak menjelaskan nama kolom dan format label. `validate_dataset_zip` membaca input sebenarnya, memeriksa arsip serta anotasi, kemudian menampilkan inventaris dan beberapa kotak ground truth untuk audit visual.

Sel split memakai `scene_id` untuk menyusun train, validation, dan test. Perhatikan jumlah gambar, scene, serta kotak tiap kelas pada `split_report`, kemudian baca isi `data.yaml`. Jika validasi atau pembagian gagal, perbaiki sumber masalah pada data; jangan melewati pemeriksaan atau menyisipkan contoh pengganti yang tidak mewakili dataset.

Sel fine-tuning menjalankan training nyata dan mencari `BEST` di direktori run. Sel berikutnya memuat bobot tersebut, menghitung metrik validation dan held-out test secara terpisah, lalu membaca `results.csv`. Putuskan konfigurasi berdasarkan validation sebelum menjalankan evaluasi test akhir. Tampilan beberapa gambar test membantu membaca jenis kesalahan, tetapi bukan pengganti metrik atau bukti bahwa seluruh kondisi baru sudah terwakili.

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 — Kontrak dataset dan upload ZIP

Siapkan ZIP berisi images/<nama>.jpg atau .png, labels/<stem>.txt, classes.json berupa list nama 1–3 kelas, manifest.csv, dan LICENSE.md. CSV wajib memiliki filename, scene_id, license, reviewed, negative. reviewed dan negative memakai true/false; label kosong hanya untuk negative=true yang sudah diperiksa. Jangan mengunggah gambar tanpa anotasi lalu menganggap label kosong sebagai ground truth.

```
from pathlib import Path, PurePosixPath
import csv, io, json, math, re, stat, zipfile, shutil, hashlib, uuid
from collections import Counter
import cv2, numpy as np, pandas as pd, matplotlib.pyplot as plt
from PIL import Image
import torch, ultralytics, yaml
from ultralytics import YOLO
from google.colab import files
from IPython.display import display
DEVICE = 0 if torch.cuda.is_available() else 'cpu'
print('Ultralytics:', ultralytics.__version__, 'device:', DEVICE)
print('Contoh struktur, bukan dataset: images/scene01_001.jpg ↔
labels/scene01_001.txt')
print('Format label: class_id x_center y_center width height
(normalized).')
print('classes.json: list nama kelas berurutan, ID mulai 0.')
print('manifest.csv header:
filename,scene_id,license,reviewed,negative')
uploaded_zip = files.upload()
if len(uploaded_zip) != 1:
    raise ValueError('Upload tepat satu ZIP dataset.')
zip_name, zip_bytes = next(iter(uploaded_zip.items()))
if Path(zip_name).suffix.lower() != '.zip' or not 0 < len(zip_bytes)
<= 100*1024**2:
    raise ValueError('ZIP wajib tidak kosong dan maksimum 100 MB.')
```

Sel 3 — Ekstraksi aman dan validasi dataset

Tidak memakai extractall. Validasi membatasi jalur, symlink, file tak dikenal, jumlah berkas, ukuran terkompresi/aktual, label dan manifest. Validasi sintaks tidak membuktikan anotasi benar secara semantik atau izin sah: pengguna tetap perlu mengaudit.

```
def validate_dataset_zip(payload, base=Path('/content')):
    if not 0 < len(payload) <= 100 * 1024**2:
        raise ValueError('Ukuran ZIP di luar batas.')
    root = Path(base) / ('dataset_raw_' + uuid.uuid4().hex[:10])
    root.mkdir(parents=True, exist_ok=False)
    max_total = 300 * 1024**2
    actual_total = 0
    try:
        with zipfile.ZipFile(io.BytesIO(payload)) as archive:
            members = archive.infolist()
            if len(members) > 1000: raise ValueError('Terlalu banyak
entri ZIP.')
            seen, declared_total = set(), 0
            for entry in members:
                name = entry.filename
                p = PurePosixPath(name)
                if (not name or '\\' in name or '\x00' in name or
p.is_absolute()
                    or ':' in name or any(part in {'..', '.'} for
part in name.split('/'))):
                    raise ValueError(f'Jalur ZIP berbahaya: {name}')
                mode = entry.external_attr >> 16
                if stat.S_ISLNK(mode) or entry.flag_bits & 1:
                    raise ValueError('Symlink atau ZIP terenkripsi
ditolak.')
                if name in seen: raise ValueError('Nama entri
duplikat.')
                seen.add(name)
                if entry.is_dir():
                    if name not in {'images/', 'labels/'}:
                        raise ValueError('Direktori ZIP tidak sesuai
kontrak.')
                    continue
                parts = p.parts
                allowed = name in
{'classes.json','manifest.csv','LICENSE.md'}
                if len(parts) == 2 and parts[0] in
{'images','labels'}:
                    if not re.fullmatch(r'[A-Za-z0-9][A-Za-z0-9_.-]
{0,100}', parts[1]):
                        raise ValueError('Nama file wajib sederhana
tanpa subdirektori.')
```

```
                    ext = p.suffix.lower()
                    allowed = ext in ({'.jpg','.jpeg','.png'} if
parts[0]=='images' else {'.txt'})
                if not allowed: raise ValueError(f'File tidak
diizinkan: {name}')
                cap = 20*1024**2 if parts[0]=='images' else
2*1024**2
                if entry.file_size > cap or entry.file_size < 0:
                    raise ValueError('File terlalu besar.')
                if entry.file_size > 1024**2 and
entry.file_size/max(1,entry.compress_size) > 200:
                    raise ValueError('Rasio kompresi mencurigakan.')
                declared_total += entry.file_size
                if declared_total > max_total: raise
ValueError('Ukuran ekspansi ZIP berlebih.')
                destination = root.joinpath(*parts).resolve()
                if root.resolve() not in destination.parents:
                    raise ValueError('Zip Slip terdeteksi.')
                destination.parent.mkdir(parents=True,exist_ok=True)
                actual_file = 0
                with archive.open(entry) as source,
destination.open('xb') as target:
                    while True:
                        block = source.read(1024*1024)
                        if not block: break
                        actual_file += len(block); actual_total +=
len(block)
                        if actual_file > cap or actual_total >
max_total:
                            raise ValueError('Ukuran aktual ekspansi
melewati batas.')
                        target.write(block)
                if actual_file != entry.file_size: raise
ValueError('Ukuran entri tidak konsisten.')
        required =
[root/'classes.json',root/'manifest.csv',root/'LICENSE.md']
        if any(not p.is_file() for p in required): raise
ValueError('Metadata wajib tidak lengkap.')
        names =
json.loads((root/'classes.json').read_text(encoding='utf-8'))
        if (not isinstance(names,list) or not 1 <= len(names) <= 3
            or any(not isinstance(n,str) or not n.strip() or
```

```
len(n)>60
                    or any(ord(ch)<32 for ch in n) for n in names)
            or len(set(names)) != len(names)):
            raise ValueError('Nama kelas harus list 1–3 nama unik
nonkosong.')
        if not (root/'LICENSE.md').read_text(encoding='utf-
8').strip():
            raise ValueError('LICENSE.md harus menjelaskan sumber
dan izin.')
        with (root/'manifest.csv').open(encoding='utf-8',newline='')
as handle:
            reader = csv.DictReader(handle)
            fields =
{'filename','scene_id','license','reviewed','negative'}
            if reader.fieldnames is None or not fields <=
set(reader.fieldnames):
                raise ValueError('Header manifest tidak lengkap.')
            records = list(reader)
        image_paths = sorted((root/'images').glob('*'))
        if not 100 <= len(image_paths) <= 200:
            raise ValueError('Validator latihan ini menerima 100–200
gambar; periksa jumlah dataset.')
        manifest_names = [r['filename'] for r in records]
        if len(set(manifest_names)) != len(records) or
set(manifest_names) != {p.name for p in image_paths}:
            raise ValueError('Manifest harus tepat satu baris per
gambar.')
        stems = [p.stem for p in image_paths]
        if len(set(stems)) != len(stems): raise ValueError('Stem
gambar harus unik.')
        if {p.name for p in (root/'labels').glob('*')} != {s+'.txt'
for s in stems}:
            raise ValueError('Label wajib berpasangan tepat dengan
gambar, tanpa label ekstra.')
        by_name = {r['filename']: r for r in records}
        pixel_hashes, inventory = set(), []
        class_totals = Counter()
        for image_path in image_paths:
            r = by_name[image_path.name]
            if (not r['scene_id'].strip() or not
r['license'].strip()
                or r['reviewed'].strip().lower() != 'true'
```

```
                or r['negative'].strip().lower() not in
{'true','false'}):
                raise ValueError('Scene/izin/review/negative belum
valid.')
            with Image.open(image_path) as header:
                if header.width * header.height > 20_000_000:
                    raise ValueError('Gambar melebihi 20
megapiksel.')
                header.verify()
            image = cv2.imread(str(image_path))
            if image is None: raise ValueError(f'Decode gagal:
{image_path.name}')
            digest =
hashlib.sha256(str(image.shape).encode()+image.tobytes()).hexdigest(
)
            if digest in pixel_hashes: raise ValueError('Gambar
duplikat identik ditolak.')
            pixel_hashes.add(digest)
            label = root/'labels'/(image_path.stem+'.txt')
            lines = [line for line in label.read_text(encoding='utf-
8').splitlines() if line.strip()]
            is_negative = r['negative'].strip().lower() == 'true'
            if bool(lines) == is_negative:
                raise ValueError('Label kosong hanya untuk negatif
yang sudah ditinjau; negatif tidak boleh memiliki bbox.')
            counts = Counter()
            for line in lines:
                tokens = line.split()
                if len(tokens)!=5 or not re.fullmatch(r'[0-
9]+',tokens[0]):
                    raise ValueError(f'Token class/label salah:
{label.name}')
                class_id = int(tokens[0])
                if not 0 <= class_id < len(names): raise
ValueError('Class ID di luar names.')
                x,y,w,h = map(float,tokens[1:])
                if (not all(math.isfinite(v) for v in (x,y,w,h))
                    or not 0 <= x <= 1 or not 0 <= y <= 1
                    or not 0 < w <= 1 or not 0 < h <= 1
                    or x-w/2 < -1e-6 or y-h/2 < -1e-6
                    or x+w/2 > 1+1e-6 or y+h/2 > 1+1e-6):
                    raise ValueError(f'Koordinat normalized tidak
```

```
valid: {label.name}')
                counts[class_id] += 1; class_totals[class_id] += 1

inventory.append({'filename':image_path.name,'scene_id':r['scene_id'
].strip(),

'license':r['license'],'negative':is_negative,

'boxes':sum(counts.values()),'class_counts':dict(counts)})
        if set(class_totals) != set(range(len(names))):
            raise ValueError('Setiap kelas wajib punya anotasi
positif.')
        if len({r['scene_id'] for r in inventory}) < 6:
            raise ValueError('Butuh minimal 6 scene independen untuk
train/val/test; disarankan 10 atau lebih.')
        return root, names, inventory
    except Exception:
        shutil.rmtree(root,ignore_errors=True)
        raise
RAW, CLASS_NAMES, inventory = validate_dataset_zip(zip_bytes)
display(pd.DataFrame([{k:v for k,v in r.items() if
k!='class_counts'} for r in inventory]).head())
print('Gambar tervalidasi:',len(inventory),'Kelas:',CLASS_NAMES)
# Audit visual anotasi: kotak ini ground truth yang diunggah, bukan
prediksi model.
for record in inventory[:3]:
    name = record['filename']
    image = cv2.imread(str(RAW/'images'/name))
    h,w = image.shape[:2]
    label_lines =
(RAW/'labels'/(Path(name).stem+'.txt')).read_text().splitlines()
    for line in label_lines:
        if not line.strip(): continue
        tokens = line.split(); idx = int(tokens[0])
        x,y,bw,bh = map(float,tokens[1:])
        left,top = round((x-bw/2)*w), round((y-bh/2)*h)
        right,bottom = round((x+bw/2)*w), round((y+bh/2)*h)
        cv2.rectangle(image,(left,top),(right,bottom),(0,255,0),2)
        cv2.putText(image,CLASS_NAMES[idx],(left,max(12,top)),
                    cv2.FONT_HERSHEY_SIMPLEX,0.5,(0,255,0),1)
    plt.figure(figsize=(9,6));
plt.imshow(cv2.cvtColor(image,cv2.COLOR_BGR2RGB))
```

```
    plt.title(f"Audit label: {name}, negative={record['negative']}")
    plt.axis('off'); plt.show()
```

Sel 4 — Split berdasarkan scene dan buat data.yaml

Seluruh scene masuk satu split. Pencarian deterministik mencoba mendekati 70/15/15 berdasarkan gambar dan memastikan tiap kelas muncul di setiap split. Jika tidak ada kandidat, tambah scene/keterwakilan kelas; jangan pecah scene untuk memperbaiki metrik.

```
import random

def split_by_scene(inventory, class_names, seed=42):
    groups = sorted({r['scene_id'] for r in inventory})
    n = len(groups)
    n_val = max(1, round(0.15*n)); n_test = max(1, round(0.15*n))
    rng = random.Random(seed)
    best = None
    for _ in range(500):
        order = groups.copy(); rng.shuffle(order)
        group_sets = {'val': set(order[:n_val]),
                      'test': set(order[n_val:n_val+n_test]),
                      'train': set(order[n_val+n_test:])}
        slices = {s:[r for r in inventory if r['scene_id'] in gs]
                  for s,gs in group_sets.items()}
        if any(not part for part in slices.values()): continue
        covered = {s:set(k for r in part for k in r['class_counts'])
                   for s,part in slices.items()}
        if any(ids != set(range(len(class_names))) for ids in
covered.values()): continue
        score = sum(abs(len(slices[s])/len(inventory)-target)
                    for s,target in [('train',0.7),('val',0.15),
('test',0.15)])
        if best is None or score < best[0]: best = (score, slices,
group_sets)
    if best is None:
        raise ValueError('Tidak ditemukan split scene dengan semua
kelas; tambah data independen.')
    _, slices, group_sets = best
    assert not (group_sets['train'] & group_sets['val'] or
group_sets['train'] & group_sets['test']
                or group_sets['val'] & group_sets['test'])
```

```
    assert sum(map(len,slices.values())) == len(inventory)
    return slices

slices = split_by_scene(inventory, CLASS_NAMES)
DATASET = Path('/content') / ('dataset_yolo_' +
uuid.uuid4().hex[:10])
for split, records in slices.items():
    image_dir = DATASET/'images'/split; label_dir =
DATASET/'labels'/split
    image_dir.mkdir(parents=True,exist_ok=True);
label_dir.mkdir(parents=True,exist_ok=True)
    for record in records:
        name = record['filename']
        shutil.copy2(RAW/'images'/name, image_dir/name)
        shutil.copy2(RAW/'labels'/(Path(name).stem+'.txt'),
label_dir/(Path(name).stem+'.txt'))
    if len(list(image_dir.glob('*'))) != len(records): raise
RuntimeError('Jumlah split salah.')
config =
{'path':str(DATASET.resolve()),'train':'images/train','val':'images/
val',
          'test':'images/test','names':dict(enumerate(CLASS_NAMES))}
DATA_YAML = DATASET/'data.yaml'
DATA_YAML.write_text(yaml.safe_dump(config,sort_keys=False,allow_uni
code=True),encoding='utf-8')
report_rows = []
for split, records in slices.items():
    totals = Counter()
    for r in records: totals.update(r['class_counts'])
    report_rows.append({'split':split,'images':len(records),
        'scenes':len({r['scene_id'] for r in records}),
        **{name:totals[i] for i,name in enumerate(CLASS_NAMES)}})
split_report = pd.DataFrame(report_rows)
display(split_report)
print(DATA_YAML.read_text())
split_report.to_csv(DATASET/'split_report.csv',index=False)
pd.DataFrame([{**{k:v for k,v in r.items() if
k!='class_counts'},'split':s}
              for s,records in slices.items() for r in
records]).to_csv(DATASET/'split_manifest.csv',index=False)
```

Sel 5 — Fine-tuning baseline

GPU disarankan; CPU sah tetapi bisa lambat. Run 30 epoch adalah awal eksperimen, bukan jaminan kualitas. Jangan menggunakan test untuk memilih threshold atau epoch.

```
run_id = 'custom_' + uuid.uuid4().hex[:8]
train_model = YOLO('yolov8n.pt')
train_results = train_model.train(data=str(DATA_YAML), epochs=30,
imgsz=640,
    batch=4, device=DEVICE, workers=2, seed=42, deterministic=True,
    patience=10, project='/content/runs_custom', name=run_id,
exist_ok=False,
    plots=True, cache=False)
RUN_DIR = Path(train_model.trainer.save_dir)
BEST = RUN_DIR/'weights'/'best.pt'
if not BEST.is_file(): raise RuntimeError('Training belum
menghasilkan best.pt.')
print('Run:',RUN_DIR,'Bobot terbaik:',BEST)
```

Sel 6 — Evaluasi validation dan test terpisah

Pilih keputusan eksperimen berdasarkan validation. Jalankan test setelah model/threshold dikunci; jangan berulang kali memilih model dari angka test.

```
custom_model = YOLO(str(BEST))
val_metrics = custom_model.val(data=str(DATA_YAML), split='val',
device=DEVICE,
                               imgsz=640, plots=True,
project=str(RUN_DIR), name='val_final')
test_metrics = custom_model.val(data=str(DATA_YAML), split='test',
device=DEVICE,
                                imgsz=640, plots=True,
project=str(RUN_DIR), name='test_final')
metric_rows = []
for split, metrics in [('val',val_metrics),('test',test_metrics)]:

metric_rows.append({'split':split,'precision_mean':float(metrics.box
.mp),
                        'recall_mean':float(metrics.box.mr),

'mAP50':float(metrics.box.map50),'mAP50_95':float(metrics.box.map)})
display(pd.DataFrame(metric_rows))
```

```
pd.DataFrame(metric_rows).to_csv(RUN_DIR/'metrics_summary.csv',index
=False)
history = pd.read_csv(RUN_DIR/'results.csv')
history.columns = history.columns.str.strip()
display(history.tail())
loss_cols = [c for c in history.columns if 'box_loss' in c]
if loss_cols:
    history.plot(x='epoch', y=loss_cols, figsize=(10,4), grid=True)
    plt.ylabel('Loss'); plt.show()
```

Sel 7 — Audit prediksi pada gambar test

Tampilkan beberapa gambar test menggunakan bobot terbaik dan kelas custom. Tidak menyatakan metrik tinggi sebelum hasil run tersedia.

```
for record in slices['test'][:5]:
    path = DATASET/'images'/'test'/record['filename']
    image = cv2.imread(str(path))
    if image is None: raise ValueError('Gambar test tidak dapat
dibaca.')
    result = custom_model.predict(image, conf=0.25, device=DEVICE,
verbose=False)[0]
    annotated = result.plot()
    plt.figure(figsize=(10,6));
plt.imshow(cv2.cvtColor(annotated,cv2.COLOR_BGR2RGB))
    plt.title(record['filename']); plt.axis('off'); plt.show()
print('Periksa bbox terhadap label yang diaudit; jangan menganggap
prediksi sebagai ground truth.')
```

Jika terjadi error

Zip Slip, symlink, ukuran berlebih atau file asing ditolak.

Ekspor ulang hanya file sesuai kontrak, tanpa path absolut, ../ atau kode di dalam arsip; jangan melemahkan validator.

Class token atau normalized bbox salah.

Periksa ekspor YOLO, ID integer 0..n-1 dan normalisasi berdasarkan lebar/tinggi asli.

Label kosong untuk gambar positif.

Lengkapi anotasi seluruh objek; negative=true hanya setelah gambar benar-benar ditinjau sebagai negatif.

Tidak ada split scene yang memenuhi semua kelas.

Tambahkan scene berkelas langka; jangan mengacak frame agar leakage terjadi.

CUDA out of memory.

Turunkan batch ke 2 atau 1 dan restart runtime jika perlu; pertahankan pencatatan parameter.

> 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.
