

## 1. Persiapan Lingkungan

Sebelum memulai, pastikan Anda telah menginstal semua dependensi yang diperlukan:

- **Python** (disarankan versi 3.8+)
- Library berikut:

```
pip install ultralytics streamlit opencv-python-headless pillow numpy
```

- **Model YOLO:** Anda bisa menggunakan model YOLO bawaan seperti yolov8n.pt, atau model kustom yang telah Anda latih.

## • 2. Struktur Proyek

- Buat direktori proyek Anda dengan struktur berikut:

```
my_yolo_app/  
|  
├─ app.py # File Streamlit utama  
├─ yolo11n.pt # Model YOLO Anda  
└─ requirements.txt # Daftar dependensi (opsional)
```

## 3. Kode Streamlit

Salin dan gunakan kode berikut untuk file app.py:

```
import streamlit as st  
import cv2  
from ultralytics import YOLO  
from PIL import Image  
import numpy as np  
from collections import Counter
```

Berikut adalah penjelasan singkat untuk setiap library yang diimpor:

1. **import streamlit as st**  
Untuk membangun antarmuka web interaktif dengan Python menggunakan framework Streamlit.
2. **import cv2**  
Untuk memproses gambar dan video secara real-time menggunakan OpenCV.
3. **from ultralytics import YOLO**  
Untuk memanfaatkan model YOLO dari library **Ultralytics** untuk deteksi objek.
4. **from PIL import Image**  
Untuk membaca dan memanipulasi gambar dalam format yang didukung oleh Pillow (seperti JPEG, PNG).
5. **import numpy as np**  
Untuk manipulasi array numerik, terutama untuk konversi gambar menjadi array numerik.

#### 6. **from collections import Counter**

Untuk menghitung jumlah kemunculan setiap objek yang terdeteksi dalam hasil deteksi.

```
@st.cache_resource
def load_model(model_path):
    return YOLO(model_path)
```

Fungsi `load_model` menggunakan dekorator `@st.cache_resource` untuk meng-cache model YOLO yang dimuat agar tidak perlu dimuat ulang setiap kali aplikasi Streamlit dijalankan atau halaman di-refresh.

#### Penjelasan:

1. **@st.cache\_resource**  
Meng-cache hasil fungsi untuk meningkatkan efisiensi. Fungsi ini hanya akan dieksekusi ulang jika parameter `model_path` berubah.
2. **def load\_model(model\_path):**  
Menerima path model YOLO sebagai input.
3. **return YOLO(model\_path)**  
Memuat model YOLO dari path yang diberikan dan mengembalikannya untuk digunakan dalam deteksi objek.

#### Manfaat:

Menghindari overhead pemuatan ulang model YOLO, sehingga aplikasi lebih cepat dan hemat sumber daya.

```
def display_results(image, results):
    boxes = results.boxes.xyxy.cpu().numpy()
    scores = results.boxes.conf.cpu().numpy()
    labels = results.boxes.cls.cpu().numpy()
    names = results.names
```

Fungsi `display_results` berfungsi untuk memproses hasil deteksi YOLO dan menggambar bounding box pada gambar asli. Berikut adalah penjelasan singkat untuk setiap baris kode:

1. **boxes = results.boxes.xyxy.cpu().numpy()**  
Mengambil koordinat bounding box dalam format `[x1, y1, x2, y2]` dari hasil deteksi dan mengonversinya menjadi array NumPy untuk manipulasi lebih lanjut.
2. **scores = results.boxes.conf.cpu().numpy()**  
Mengambil skor kepercayaan (confidence scores) untuk setiap bounding box, yang menunjukkan seberapa yakin model terhadap deteksi tersebut.
3. **labels = results.boxes.cls.cpu().numpy()**  
Mengambil indeks kelas dari setiap objek yang terdeteksi, mewakili kategori objek.

#### 4. **names = results.names**

Mengambil nama-nama kelas objek dari model YOLO, yang akan digunakan untuk mengonversi indeks kelas (labels) menjadi nama kelas (misalnya, "person", "car").

#### **Kesimpulan:**

Bagian ini mempersiapkan data deteksi (bounding box, confidence score, dan label) untuk divisualisasikan pada gambar.

```
detected_objects = []
for i in range(len(boxes)):
    if scores[i] > 0.5: # Confidence threshold
        x1, y1, x2, y2 = boxes[i].astype(int)
        label = names[int(labels[i])]
        score = scores[i]
        detected_objects.append(label)
        cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)
        cv2.putText(image, f"{label}: {score:.2f}", (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX,
return image, detected_objects
```

Berikut adalah penjelasan ringkas dari potongan kode ini:

#### **Fungsi Utama:**

Kode ini menganalisis hasil deteksi objek, memfilter berdasarkan confidence threshold, menggambar bounding box pada gambar, dan menyimpan label objek yang terdeteksi.

---

#### **Penjelasan Tiap Baris:**

1. **detected\_objects = []**  
Inisialisasi daftar kosong untuk menyimpan nama objek yang terdeteksi.
2. **for i in range(len(boxes)):**  
Iterasi melalui setiap bounding box yang terdeteksi.
3. **if scores[i] > 0.5:**  
Memfilter hasil deteksi dengan confidence score lebih dari 0.5 agar hanya deteksi yang cukup akurat yang digunakan.
4. **x1, y1, x2, y2 = boxes[i].astype(int)**  
Mengambil koordinat bounding box dan mengonversinya ke bilangan bulat.
5. **label = names[int(labels[i])]**  
Mendapatkan nama kelas objek dari indeks kelas (labels).
6. **score = scores[i]**  
Mendapatkan nilai confidence score untuk deteksi ini.
7. **detected\_objects.append(label)**  
Menambahkan nama objek yang terdeteksi ke daftar detected\_objects.
8. **cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)**  
Menggambar bounding box hijau di sekitar objek pada gambar.

9. `cv2.putText(image, f"{label}: {score:.2f}", (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)`  
Menambahkan label dan confidence score sebagai teks di atas bounding box.
  10. `return image, detected_objects`  
Mengembalikan gambar yang telah diberi anotasi dan daftar nama objek yang terdeteksi.
- 

#### Hasil Akhir:

- **image:** Gambar dengan bounding box dan label yang divisualisasikan.
- **detected\_objects:** Daftar objek yang terdeteksi, misalnya ["person", "car"].

```
def display_results(image, results):  
    boxes = results.boxes.xyxy.cpu().numpy()  
    scores = results.boxes.conf.cpu().numpy()  
    labels = results.boxes.cls.cpu().numpy()  
    names = results.names
```

Berikut adalah penjelasan singkat untuk fungsi main:

#### Tujuan Fungsi:

Mengatur antarmuka aplikasi Streamlit untuk melakukan deteksi objek dengan model YOLO.

---

#### Penjelasan Tiap Baris:

1. `st.title("Deteksi Objek dengan YOLO")`  
Menampilkan judul utama pada halaman aplikasi Streamlit.
  2. `st.sidebar.title("Pengaturan")`  
Menambahkan bagian sidebar dengan judul "Pengaturan" untuk menyediakan kontrol dan opsi konfigurasi.
  3. `model_path = "yolo11n.pt"`  
Menentukan path ke file model YOLO yang akan digunakan untuk deteksi objek.
  4. `model = load_model(model_path)`  
Memanggil fungsi `load_model` untuk memuat model YOLO dari path yang telah ditentukan, menggunakan caching untuk meningkatkan efisiensi.
- 

#### Fungsi Ini Selanjutnya:

Fungsi main adalah titik awal aplikasi Streamlit. Setelah bagian ini, Anda dapat menambahkan fitur seperti pemilihan mode deteksi (gambar, video, atau kamera) dan menjalankan deteksi objek berdasarkan input pengguna.

```
mode = st.sidebar.radio("Pilih Mode", ["Gambar", "Video", "Kamera"])
```

Baris kode ini berfungsi untuk menyediakan opsi bagi pengguna untuk memilih mode deteksi objek melalui **Streamlit sidebar**.

---

#### Penjelasan:

1. **st.sidebar.radio**  
Membuat kontrol radio button di sidebar Streamlit, di mana pengguna hanya dapat memilih satu opsi dari beberapa pilihan.
  2. **"Pilih Mode"**  
Label yang ditampilkan di sebelah kontrol radio button.
  3. **["Gambar", "Video", "Kamera"]**  
Daftar opsi yang dapat dipilih pengguna:
    - **"Gambar"**: Untuk deteksi pada gambar yang diunggah.
    - **"Video"**: Untuk deteksi pada video yang diunggah.
    - **"Kamera"**: Untuk deteksi secara real-time menggunakan kamera.
  4. **mode = ...**  
Variabel mode akan menyimpan opsi yang dipilih pengguna. Nilainya bisa berupa "Gambar", "Video", atau "Kamera", tergantung pada input pengguna.
- 

#### Fungsi:

Kode ini menentukan alur kerja aplikasi berdasarkan pilihan pengguna:

- Jika pengguna memilih **Gambar**, aplikasi akan mengaktifkan modul deteksi gambar.
- Jika memilih **Video**, aplikasi akan menjalankan deteksi pada video.
- Jika memilih **Kamera**, aplikasi akan mengaktifkan deteksi real-time melalui kamera.

```
if mode == "Gambar":
    uploaded_file = st.file_uploader("Unggah Gambar", type=["jpg", "png", "jpeg"])
    if uploaded_file:
        image = Image.open(uploaded_file)
        image = np.array(image)
        results = model.predict(image, imgsz=640)
        image, detected_objects = display_results(image, results[0])
        st.image(image, caption="Hasil Deteksi", channels="RGB", use_column_width=True)
```

#### Penjelasan Singkat:

Kode ini menangani mode **deteksi objek pada gambar** berdasarkan input dari pengguna.

---

#### Penjelasan Tiap Baris:

1. **if mode == "Gambar":**  
Mengecek apakah pengguna telah memilih mode deteksi "**Gambar**" dari radio button di sidebar.
2. **uploaded\_file = st.file\_uploader("Unggah Gambar", type=["jpg", "png", "jpeg"])**  
Menyediakan widget untuk mengunggah file gambar dengan format yang didukung (JPG, PNG, JPEG).
3. **if uploaded\_file:**  
Mengecek apakah pengguna telah mengunggah sebuah file.
4. **image = Image.open(uploaded\_file)**  
Membuka file gambar yang diunggah menggunakan **Pillow**.
5. **image = np.array(image)**  
Mengonversi gambar menjadi array NumPy untuk keperluan pemrosesan.
6. **results = model.predict(image, imsz=640)**  
Menggunakan model YOLO untuk mendeteksi objek pada gambar dengan ukuran input 640 piksel.
7. **image, detected\_objects = display\_results(image, results[0])**  
Memproses hasil deteksi untuk menggambar bounding box dan mendapatkan daftar objek yang terdeteksi menggunakan fungsi `display_results`.
8. **st.image(image, caption="Hasil Deteksi", channels="RGB", use\_column\_width=True)**  
Menampilkan gambar dengan bounding box yang sudah digambar bersama dengan teks deskriptif ("Hasil Deteksi").

---

#### Hasil Akhir:

- **Input:** Pengguna mengunggah file gambar.
  - **Proses:** YOLO mendeteksi objek dalam gambar.
- Output:** Gambar dengan bounding box ditampilkan di halaman aplikasi bersama label deteksi.

```

elif mode == "Video":
    uploaded_file = st.file_uploader("Unggah Video", type=["mp4", "avi", "m
    if uploaded_file:
        tfile = uploaded_file.name
        with open(tfile, "wb") as f:
            f.write(uploaded_file.read())
        cap = cv2.VideoCapture(tfile)
        st_frame = st.empty()
        while cap.isOpened():
            ret, frame = cap.read()
            if not ret:
                break
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            results = model.predict(frame, imgsz=640)
            frame, _ = display_results(frame, results[0])
            st_frame.image(frame, channels=3, use_column_width=True)
        cap.release()

```

### Penjelasan Kode:

Kode ini menangani mode **deteksi objek pada video** yang diunggah pengguna.

---

### Penjelasan Tiap Baris:


- elif mode == "Video":**  
Mengecek apakah pengguna telah memilih mode deteksi "Video".
- uploaded\_file = st.file\_uploader("Unggah Video", type=["mp4", "avi", "mov"])**  
Menyediakan widget untuk mengunggah file video dengan format yang didukung (MP4, AVI, MOV).
- if uploaded\_file:**  
Mengecek apakah pengguna telah mengunggah sebuah file video.
- tfile = uploaded\_file.name**  
Menggambil nama file dari video yang diunggah.
- with open(tfile, "wb") as f:**  
Membuka file lokal dengan nama yang sama dan dalam mode tulis biner (wb).
- f.write(uploaded\_file.read())**  
Menulis konten video yang diunggah ke file lokal.
- cap = cv2.VideoCapture(tfile)**  
Membuka file video menggunakan OpenCV untuk diproses frame per frame.
- st\_frame = st.empty()**  
Membuat placeholder di Streamlit untuk menampilkan video yang diproses.
- while cap.isOpened():**  
Memulai loop untuk membaca frame demi frame dari video hingga selesai.
- ret, frame = cap.read()**  
Membaca frame dari video. ret akan False jika tidak ada lagi frame yang tersedia.
- if not ret: break**  
Menghentikan loop jika video selesai diproses.

12. **frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)**  
Mengonversi frame dari format BGR (OpenCV default) ke RGB untuk ditampilkan di Streamlit.
  13. **results = model.predict(frame, imgsz=640)**  
Melakukan deteksi objek pada frame dengan menggunakan model YOLO.
  14. **frame, \_ = display\_results(frame, results[0])**  
Memproses hasil deteksi untuk menggambar bounding box pada frame.
  15. **st\_frame.image(frame, channels="RGB", use\_column\_width=True)**  
Menampilkan frame yang telah diberi bounding box ke halaman aplikasi secara real-time.
  16. **cap.release()**  
Menutup file video setelah semua frame selesai diproses.
- 

#### Hasil Akhir:

- **Input:** Pengguna mengunggah file video.
- **Proses:** Setiap frame dalam video diproses oleh YOLO untuk mendeteksi objek.
- **Output:** Video diputar di halaman aplikasi dengan bounding box pada objek yang terdeteksi.

```
if mode == "Kamera":
    run_detection = st.checkbox("Mulai Deteksi Kamera")
    if run_detection:
        cap = cv2.VideoCapture(0)
        st_frame = st.empty()
        while True:
            ret, frame = cap.read()
            if not ret:
                st.warning("Gagal menangkap frame.")
                break
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            results = model.predict(frame, imgsz=640)
            frame, detected_objects = display_results(frame, results[0])
            st_frame.image(frame, channels="RGB", use_column_width=True)
            if not st.session_state.get("run_detection", True):
                break
        cap.release()
```



#### Penjelasan Kode:

Kode ini menangani mode **deteksi objek secara real-time melalui kamera**.

---

#### Penjelasan Tiap Baris:

1. **elif mode == "Kamera":**  
Mengecek apakah pengguna telah memilih mode deteksi "**Kamera**".
2. **run\_detection = st.checkbox("Mulai Deteksi Kamera")**  
Menyediakan checkbox di halaman aplikasi Streamlit untuk memulai atau menghentikan deteksi kamera.



3. **if run\_detection:**  
Mengecek apakah pengguna telah mengaktifkan checkbox untuk memulai deteksi.
  4. **cap = cv2.VideoCapture(0)**  
Membuka koneksi ke kamera default (indeks 0) menggunakan OpenCV.
  5. **st\_frame = st.empty()**  
Membuat placeholder di Streamlit untuk menampilkan video dari kamera secara real-time.
  6. **while True:**  
Memulai loop untuk menangkap dan memproses frame dari kamera secara terus-menerus.
  7. **ret, frame = cap.read()**  
Membaca frame dari kamera. ret akan False jika frame gagal ditangkap.
  8. **if not ret:**  
Mengecek apakah frame berhasil ditangkap. Jika tidak, menampilkan peringatan dan menghentikan loop.
  9. **st.warning("Gagal menangkap frame.")**  
Menampilkan peringatan di aplikasi jika kamera gagal menangkap frame.
  10. **frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)**  
Mengonversi frame dari format BGR (OpenCV default) ke RGB untuk ditampilkan di Streamlit.
  11. **results = model.predict(frame, imgsz=640)**  
Melakukan deteksi objek pada frame dengan menggunakan model YOLO.
  12. **frame, detected\_objects = display\_results(frame, results[0])**  
Memproses hasil deteksi untuk menggambar bounding box pada frame dan mengumpulkan daftar objek yang terdeteksi.
  13. **st\_frame.image(frame, channels="RGB", use\_column\_width=True)**  
Menampilkan frame yang telah diberi bounding box ke halaman aplikasi secara real-time.
  14. **if not st.session\_state.get("run\_detection", True):**  
Mengecek apakah checkbox telah dinonaktifkan oleh pengguna. Jika ya, menghentikan loop.
  15. **break**  
Keluar dari loop jika checkbox dinonaktifkan.
  16. **cap.release()**  
Menutup koneksi ke kamera setelah proses deteksi selesai.
- 

#### Hasil Akhir:

- **Input:** Kamera menangkap video secara real-time.
- **Proses:** Frame dari kamera diproses oleh YOLO untuk mendeteksi objek.
- **Output:** Video kamera ditampilkan di halaman aplikasi dengan bounding box pada objek yang terdeteksi.

```
if __name__ == "__main__":  
    main()
```

Baris ini adalah titik masuk utama untuk menjalankan aplikasi Python. Berikut adalah penjelasan singkat:

---

**if \_\_name\_\_ == "\_\_main\_\_":**

- **Fungsi Utama:** Memastikan bahwa kode dalam blok ini hanya dijalankan jika file Python sedang dieksekusi secara langsung (bukan diimpor sebagai modul).
  - Saat file dijalankan sebagai skrip utama, nilai `__name__` adalah `"__main__"`.
  - Jika file diimpor, nilai `__name__` akan berupa nama modul (nama file tanpa ekstensi).
- 

**main()**

- Memanggil fungsi `main` untuk menjalankan logika utama aplikasi Streamlit, termasuk menampilkan antarmuka pengguna dan memproses input.
- 

#### Tujuan:

Kode ini memastikan bahwa fungsi `main()` hanya akan dipanggil ketika file dijalankan langsung, sehingga mencegah eksekusi yang tidak diinginkan jika file ini digunakan sebagai modul oleh skrip lain.

#### 4. Menjalankan Aplikasi Secara Lokal

1. Pastikan Anda berada di direktori proyek:

```
cd my_yolo_app
```

2. Jalankan aplikasi Streamlit

```
streamlit run app.py
```

3. Akses aplikasi di browser melalui URL yang diberikan, biasanya <http://localhost:8501>.

#### 5. Deploy ke Cloud (Streamlit Community Cloud)

1. Buat akun di [Streamlit Community Cloud](#).
2. Fork atau unggah proyek Anda ke repositori GitHub.
3. Di Streamlit Cloud, pilih "Deploy an App" dan hubungkan dengan repositori GitHub Anda.

4. Masukkan path app.py sebagai file utama.
5. Klik **Deploy**.

