

Nama : Yilita Oktavia

Nim : 230741106

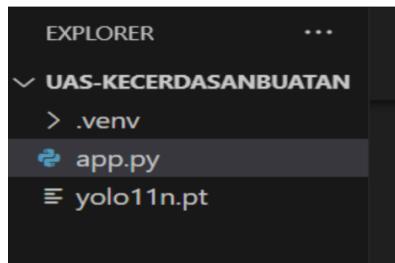
Prodi : Ilmu Komputer

Makul : Kecerdasan Buatan / artificial intelligent (AI)

Fakultas : Teknik Dan sain

PENJELASAN TUGAS UAS

Langkah Pertama



1. Buat Folder Proyek

- Buat folder baru untuk proyek, misalnya: UAS-KECERDASANBUATAN
Setelah folder baru terbuat masukan file Yollo 11
- Di dalam folder ini, buat file utama file.py untuk menjalankan kode Python.

Langka Kedua

Codingan

```
file.py > ...
1  from ultralytics import YOLO
2  import cv2
3  import streamlit as st
4  from PIL import Image
5  import numpy as np
6  from collections import Counter
7
```

1. from ultralytics import YOLO

- Mengimpor kelas YOLO dari pustaka Ultralytics.
- Kelas ini digunakan untuk menjalankan model YOLO (You Only Look Once) untuk deteksi objek.

2. import cv2

- Mengimpor pustaka OpenCV.

- OpenCV digunakan untuk pemrosesan gambar dan video, seperti membaca, menampilkan, dan memanipulasi data visual.
3. import streamlit as st
 - Mengimpor pustaka Streamlit dengan alias st.
 - Streamlit digunakan untuk membuat antarmuka web berbasis Python, mempermudah pembuatan aplikasi untuk menampilkan hasil deteksi.
 4. from PIL import Image
 - Mengimpor kelas Image dari pustaka Pillow (PIL).
 - Image digunakan untuk memuat dan memproses gambar, seperti membuka file gambar untuk ditampilkan atau dimanipulasi.
 5. import numpy as np
 - Mengimpor pustaka NumPy dengan alias np.
 - NumPy digunakan untuk manipulasi array atau matriks, yang sering digunakan dalam pemrosesan gambar.
 6. from collections import Counter
 - Mengimpor Counter dari pustaka bawaan Python, collections.
 - Counter digunakan untuk menghitung jumlah elemen dalam sebuah iterable, misalnya menghitung jumlah deteksi objek berdasarkan label.

Kesimpulan

Kode ini adalah bagian awal dari program Python untuk deteksi objek.

- Menggabungkan model YOLO (untuk deteksi objek), OpenCV (pemrosesan gambar/video), dan Streamlit (antarmuka web) dengan alat bantu seperti Pillow (manipulasi gambar) dan NumPy (manipulasi data).
- Counter kemungkinan akan digunakan untuk menghitung jumlah objek tertentu yang terdeteksi.

Langkah ketiga

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

1. Komentar: # Load YOLO model

- Memberikan penjelasan bahwa kode ini bertujuan untuk memuat model YOLO.

2. Dekorator: `@st.cache_resource`

- Penjelasan:
 - Merupakan fitur dari Streamlit untuk melakukan caching sumber daya.
 - Fungsi yang didekorasi dengan `@st.cache_resource` akan menyimpan hasilnya di cache. Jika fungsi ini dipanggil lagi dengan parameter yang sama, hasil yang sudah di-cache akan digunakan tanpa memuat ulang model.
- Tujuan:
 - Menghemat waktu dan sumber daya dengan mencegah pemuat ulang model YOLO setiap kali halaman Streamlit di-refresh.

3. Fungsi: `load_model(model_path)`

- Penjelasan:
 - Fungsi ini bertujuan untuk memuat model YOLO dari path model yang diberikan.
- Parameter:
 - `model_path`: Path (lokasi file) model YOLO yang akan dimuat, misalnya `yolo11n.pt`.
- Isi Fungsi:
 - Baris return `YOLO(model_path)` memuat model YOLO menggunakan pustaka Ultralytics.

Cara Kerja:

- Ketika fungsi `load_model(model_path)` dipanggil:
 1. Streamlit memeriksa apakah model dengan `model_path` sudah ada di cache.
 2. Jika belum ada di cache, model akan dimuat menggunakan `YOLO(model_path)`, dan hasilnya disimpan di cache.
 3. Jika sudah ada di cache, Streamlit langsung mengambil model dari cache tanpa memuat ulang.

Tujuan Utama:

- Memuat model YOLO hanya sekali, sehingga meningkatkan performa aplikasi (tidak mengulang proses yang berat).
- Menggunakan dekorator `@st.cache_resource` untuk memanfaatkan efisiensi caching Streamlit.

Langkah Keempat

```
# Process and display the detection results
def display_results(image, results):
    boxes = results.bboxes.xyxy.cpu().numpy() # [x1, y1, x2, y2]
    scores = results.bboxes.conf.cpu().numpy() # Confidence scores
    labels = results.bboxes.cls.cpu().numpy() # Class indices
    names = results.names # Class names

    detected_objects = []
```

Fungsi `display_results` memproses hasil deteksi objek dari model, menggambar kotak batas (bounding box) dan label di atas gambar, lalu menampilkan gambar dengan deteksi tersebut.

Langkah singkatnya:

1. Ekstrak hasil deteksi:
 - o boxes: Koordinat bounding box.
 - o scores: Skor kepercayaan model.
 - o labels: Indeks kelas objek.
 - o names: Nama-nama kelas.
2. Simpan informasi deteksi: Membuat list objek yang terdeteksi, termasuk nama kelas, skor kepercayaan, dan koordinat bounding box.
3. Gambar pada gambar:
 - o Menggambar kotak hijau di sekitar objek.
 - o Menambahkan teks label (nama kelas dan skor kepercayaan).
4. Tampilkan gambar: Menampilkan gambar hasil deteksi menggunakan OpenCV.
5. Kembalikan hasil: Mengembalikan list objek terdeteksi.

Contoh output list:

python

Copy code

[

```
{"class_name": "person", "confidence": 0.95, "bounding_box": [50, 30, 200, 400]}
```

]

Langkah kelima

```
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, 0.5, (0, 255, 0), 2)

return image, detected_objects
```

Kode ini **memfilter dan menggambar deteksi objek** pada gambar berdasarkan ambang batas kepercayaan ($\text{scores}[i] > 0.5$). Berikut penjelasan langkahnya:

1. Iterasi melalui semua deteksi:

Menggunakan `for i in range(len(boxes))` untuk memproses setiap deteksi.

2. Filter confidence score:

Hanya deteksi dengan skor kepercayaan > 0.5 yang diproses.

3. Ekstrak data deteksi:

- o Koordinat bounding box ($x1, y1, x2, y2$) diubah menjadi integer.
- o Label kelas diambil dari `names` menggunakan indeks `labels[i]`.
- o Confidence score disimpan dalam `score`.

4. Tambahkan ke list deteksi:

Nama kelas objek (`label`) ditambahkan ke list `detected_objects`.

5. Gambar pada gambar:

- o Gambar **kotak hijau** di sekitar objek menggunakan `cv2.rectangle`.
- o Tambahkan **teks label dan skor kepercayaan** di atas kotak menggunakan `cv2.putText`.

Tujuannya: Menampilkan hanya deteksi yang valid ($\text{confidence} > 0.5$) dan menggambarkannya pada gambar.

```
# Main Streamlit app
def main():
    st.title("Real-time Object Detection with YOLO")
    st.sidebar.title("Settings")

    model_path = "yolo1in.pt" # Path to your YOLO model
    model = load_model(model_path)

    # Create the checkbox once
    run_detection = st.sidebar.checkbox("Start/Stop Object Detection", key="detection_control")
```

Kode di atas adalah bagian dari aplikasi **Streamlit** untuk deteksi objek secara real-time menggunakan YOLO. Berikut penjelasan singkatnya:

1. Judul Aplikasi:

st.title("Real-time Object Detection with YOLO") menampilkan judul utama aplikasi di halaman.

2. Sidebar untuk Pengaturan:

- st.sidebar.title("Settings") menambahkan judul bagian di sidebar.
- st.sidebar.checkbox("Start/Stop Object Detection", key="detection_control") membuat checkbox untuk menghidupkan/mematikan deteksi objek.

3. Model YOLO:

- model_path adalah lokasi file model YOLO (yolo11n.pt).
- model = load_model(model_path) memuat model YOLO yang akan digunakan.

Tujuan:

Membuat antarmuka untuk mengatur dan menjalankan deteksi objek secara real-time dengan kontrol di sidebar.

```
# Open video capture if checkbox is active
if run_detection:
    cap = cv2.VideoCapture(0)
    st_frame = st.empty() # Placeholder for video frames
    st_detection_info = st.empty() # Placeholder for detection information

    while True:
        ret, frame = cap.read()
        if not ret:
            st.warning("Failed to capture image.")
            break
```

Kode di atas membuka kamera untuk deteksi objek real-time jika checkbox aktif:

1. Membuka Kamera:

cv2.VideoCapture(0) membuka kamera untuk menangkap video.

2. Placeholder Streamlit:

st.empty() digunakan untuk menampilkan frame video dan informasi deteksi dalam aplikasi.

3. Loop untuk Membaca Frame:

Selama deteksi aktif, aplikasi membaca frame dari kamera dengan cap.read(). Jika gagal membaca frame, aplikasi memberi peringatan dan berhenti.

Tujuan: Menangkap dan menampilkan video serta informasi deteksi objek di aplikasi Streamlit.

```

# Run YOLO detection
frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB) # Convert to RGB for display
results = model.predict(frame, imgsz=640) # Perform detection

# Draw results and collect detected objects
frame, detected_objects = display_results(frame, results[0])

# Display video feed
st_frame.image(frame, channels="RGB", use_column_width=True)

```

Kode di atas menjalankan deteksi objek YOLO dan menampilkan hasilnya dalam aplikasi Streamlit:

- Mengonversi Warna:**

cv2.cvtColor(frame, cv2.COLOR_BGR2RGB) mengonversi frame dari format BGR (standar OpenCV) ke RGB (format yang digunakan oleh Streamlit).

- Deteksi dengan YOLO:**

model.predict(frame, imgsz=640) menjalankan deteksi objek menggunakan model YOLO dengan ukuran gambar input 640x640.

- Menggambar Hasil Deteksi:**

frame, detected_objects = display_results(frame, results[0]) menggambar kotak deteksi objek dan mengembalikan gambar beserta objek yang terdeteksi.

- Menampilkan Video:**

st_frame.image(frame, channels="RGB", use_column_width=True) menampilkan frame video yang telah diproses di aplikasi Streamlit.

Tujuan: Menangkap frame, melakukan deteksi objek, menggambar hasil deteksi, dan menampilkan video secara real-time.

```

# Display detection information
if detected_objects:
    object_counts = Counter(detected_objects)
    detection_info = "\n".join([f"{obj}: {count}" for obj, count in object_counts.items()])
else:
    detection_info = "No objects detected."

st_detection_info.text(detection_info) # Update detection info text

# Break the loop if checkbox is unchecked
if not st.session_state.detection_control:
    break

cap.release()

if __name__ == "__main__":
    main()

```

Kode ini menampilkan informasi deteksi objek dan menghentikan loop jika checkbox dimatikan:

1. Menghitung Objek Terdeteksi:

Counter(detected_objects) menghitung jumlah setiap objek yang terdeteksi dan menyimpannya dalam format yang mudah dibaca.

2. Menampilkan Informasi Deteksi:

Jika ada objek yang terdeteksi, informasi deteksi ditampilkan dengan st_detection_info.text(detection_info). Jika tidak ada objek yang terdeteksi, akan ditampilkan pesan "No objects detected."

3. Hentikan Loop:

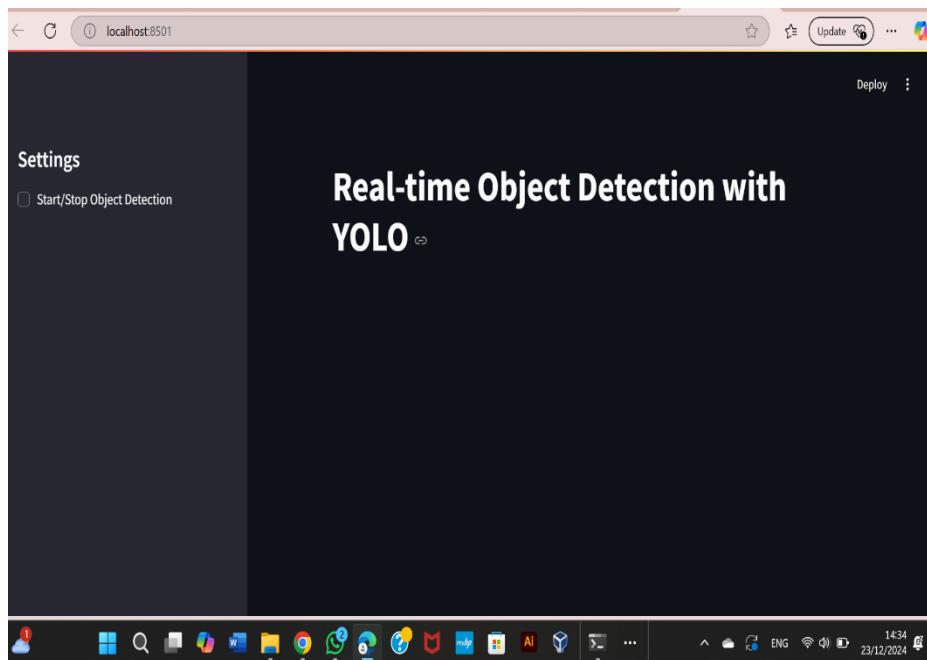
Jika checkbox deteksi dimatikan (st.session_state.detection_control bernilai False), loop akan berhenti.

4. Tutup Kamera:

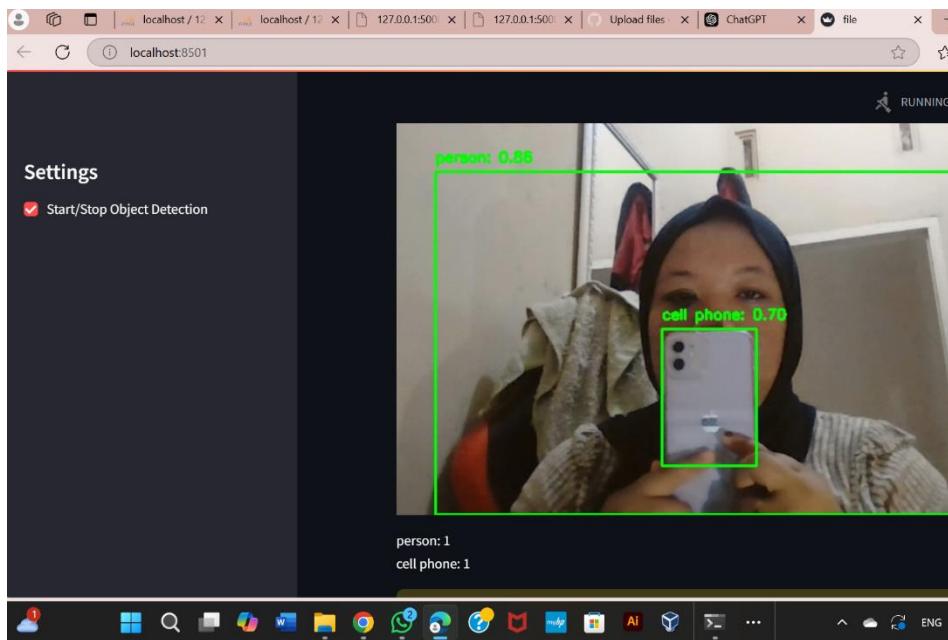
cap.release() menutup kamera setelah loop selesai.

Tujuan: Menampilkan jumlah deteksi objek secara real-time dan menghentikan deteksi saat checkbox dimatikan.

Setelah codingan telah di isi di visual code lalu kita jalankan kan hasilnya seperti ini



Lalu jika kita klik star/stop hasilnya akan begini



INI SAJA TUTORIAL NYA SELAMAT MENCUBA 😊