supervision

democratising computer vision • powered by roboflow

Onuralp SEZER
Supervision Co-Maintainer
Senior Software Engineer

Supervision Nedir ?

Supervision, yeniden kullanılabilir bilgisayarlı görüntü işleme araçları sunar. Veri kümenizi sabit diskinizden yüklemeniz veya görüntü,video üzerinde tespitleri çizmeniz veya bir bölgedeki tespit sayısını sayma işlemlerini yapabilirsiniz.

Supervision bağımsız olacak şekilde tasarlanmıştır. Sadece herhangi bir sınıflandırma, tespit veya segmentasyon modelini takın. Kolaylık için, Ultralytics, Transformers veya MMDetection gibi en popüler kütüphaneler için bağlayıcılar oluşturduk.

Detections

- sv.Detections, çeşitli nesne tespiti ve segmentasyon modellerinden gelen sonuçları tek, birleştirilmiş bir formata dönüştürür.
- Veri manipülasyonunu ve filtrelemeyi kolaylaştırır.
- “tracking”, “annotators” ve “Zone” araçları için tutarlı bir API sağlar.

```python
import cv2
import supervision as sv
from ultralytics import YOLO

image = cv2.imread(<SOURCE_IMAGE_PATH>)
model = YOLO('yolov8s.pt')
annotator = sv.BoundingBoxAnnotator()

result = model(image)[0]
detections = sv.Detections.from_ultralytics(result)

annotated_image = annotator.annotate(image,
detections)
```

https://supervision.roboflow.com/latest/detection/core/

Classifications

```python
import torch
import clip
import cv2
from PIL import Image
import supervision as sv


device = "cuda" if torch.cuda.is_available() else "cpu"

model, preprocess = clip.load("ViT-B/32", device=device)
image = preprocess(Image.open("bus.jpg")).unsqueeze(0).to(device)

text = clip.tokenize(["a bus", "a cake", "a stop sign"]).to(device)
output, _ = model(image, text)
classifications = sv.Classifications.from_clip(output)

tokens = ["bus", "cake", "stop sign"]
for i in range(len(tokens)):
  print(f"The model predicted the image was a {tokens[i]}"
   "with a confidence of {classifications.confidence[i]:.2f}")
```

- The model predicted the image was a bus with a confidence of 1.00
- The model predicted the image was a cake with a confidence of 0.00
- The model predicted the image was a stop sign with a confidence of 0.00

https://supervision.roboflow.com/latest/classification/core/

Annotators

- Supervision Annotators ile, görüntüler üzerinde çeşitli nesne tespiti veya segmentasyon modellerinin çıktılarını işaretlemek ve görsel olarak vurgulamak için kullanılır.

- Örneğin, bir nesne tespit modelinin çıktılarını annotate etmek için sv.BoundingBoxAnnotator ve sv.LabelAnnotator sınıfları kullanılır. Eğer segmentasyon modeli çalıştırıyorsanız, sv.MaskAnnotator kullanabiliriz.

- Annotator'ler, görsel sonuçları yorumlama, analiz etme ve paylaşma süreçlerinde kullanıcıya yardımcı olmak için tasarlanmıştır.

https://supervision.roboflow.com/latest/annotators/

Annotators

```python
import cv2
import supervision as sv
from inference.models.utils import get_roboflow_model

model = get_roboflow_model(model_id="yolov8n-640",
                           api_key=
<ROBOFLOW_API_KEY>)
image = cv2.imread(<PATH TO IMAGE>)

results = model.infer(image)[0]

detections = sv.Detections.from_inference(results)

bounding_box_annotator = sv.BoundingBoxAnnotator()
label_annotator = sv.LabelAnnotator()

annotated_image = bounding_box_annotator.annotate(
    scene=image, detections=detections)
annotated_image = label_annotator.annotate(
    scene=annotated_image, detections=detections)
```

https://supervision.roboflow.com/latest/annotators/

Track Objects

```python
import numpy as np
import supervision as sv
from ultralytics import YOLO

model = YOLO("yolov8n.pt")
tracker = sv.ByteTrack()
box_annotator = sv.BoundingBoxAnnotator()
label_annotator = sv.LabelAnnotator()
trace_annotator = sv.TraceAnnotator()

def callback(frame: np.ndarray, _: int) -> np.ndarray:
    results = model(frame)[0]
    detections = sv.Detections.from_ultralytics(results)
    detections = tracker.update_with_detections(detections)

    labels = [
        f"#{tracker_id} {results.names[class_id]}"
        for class_id, tracker_id
        in zip(detections.class_id, detections.tracker_id)
    ]

    annotated_frame = box_annotator.annotate(
        frame.copy(), detections=detections)
    annotated_frame = label_annotator.annotate(
        annotated_frame, detections=detections, labels=labels)
    return trace_annotator.annotate(
        annotated_frame, detections=detections)

sv.process_video(
    source_path="people-walking.mp4",
    target_path="result.mp4",
    callback=callback
)
```

https://supervision.roboflow.com/latest/how_to/track_objects/#annotate-video-with-traces

Track Objects

```python
import numpy as np
import supervision as sv
from ultralytics import YOLO

model = YOLO("yolov8n.pt")
tracker = sv.ByteTrack()
box_annotator = sv.BoundingBoxAnnotator()
label_annotator = sv.LabelAnnotator()
trace_annotator = sv.TraceAnnotator()

def callback(frame: np.ndarray, _: int) -> np.ndarray:
    results = model(frame)[0]
    detections = sv.Detections.from_ultralytics(results)
    detections = tracker.update_with_detections(detections)

    labels = [
        f"#{tracker_id} {results.names[class_id]}"
        for class_id, tracker_id
        in zip(detections.class_id, detections.tracker_id)
    ]

    annotated_frame = box_annotator.annotate(
        frame.copy(), detections=detections)
    annotated_frame = label_annotator.annotate(
        annotated_frame, detections=detections, labels=labels)
    return trace_annotator.annotate(
        annotated_frame, detections=detections)

sv.process_video(
    source_path="people-walking.mp4",
    target_path="result.mp4",
    callback=callback
)
```

https://supervision.roboflow.com/latest/how_to/track_objects/#annotate-video-with-traces

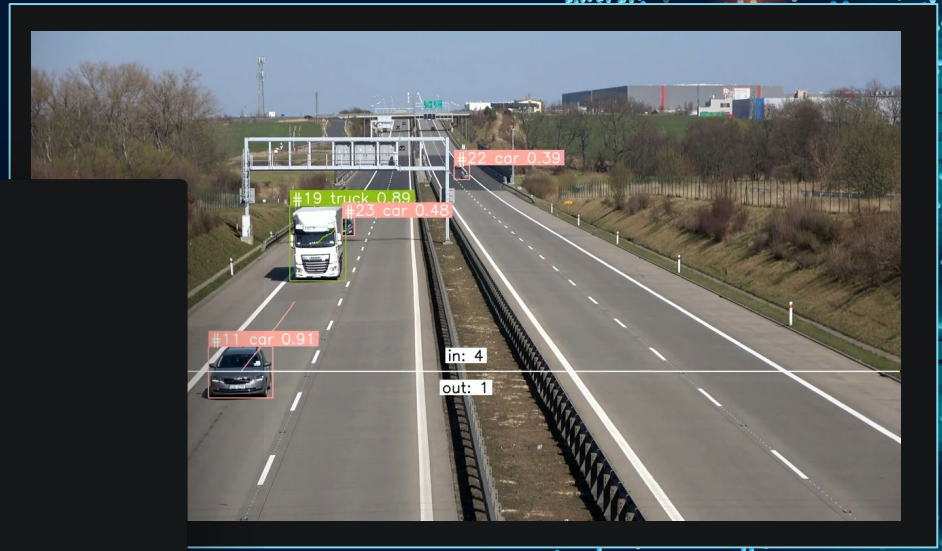
Line Zone

```python
import supervision as sv
from ultralytics import YOLO

model = YOLO(<SOURCE_MODEL_PATH>)
tracker = sv.ByteTrack()
frames_generator = sv.get_video_frames_generator(<SOURCE_VIDEO_PATH>)
start, end = sv.Point(x=0, y=1080), sv.Point(x=3840, y=1080)
line_zone = sv.LineZone(start=start, end=end)

for frame in frames_generator:
    result = model(frame)[0]
    detections = sv.Detections.from_ultralytics(result)
    detections = tracker.update_with_detections(detections)
    crossed_in, crossed_out = line_zone.trigger(detections)

line_zone.in_count, line_zone.out_count
# 7, 2
```



https://supervision.roboflow.com/latest/detection/tools/line_zone/

supervision

$ hackfest

Build, learn, and contribute to
Supervision open-source vision tools.

</> Contribute

Teşekkürler!

Sorusu olan varsa dinleyelim ? :)

Onuralp SEZER - @onuralpszr

Github: https://github.com/onuralpszr

Linkedin: https://www.linkedin.com/in/onuralpszr

Supervision: https://supervision.roboflow.com/