Progress Documentation

1. Initial YOLOv8 Object Detection Implementation

Overview

This program tests YOLOv8 for object detection on a single image, verifying the model's functionality and output capabilities.

Key Features

1. **Model Loading**
    - Uses pre-trained YOLOv8n weights (yolov8n.pt) for lightweight and efficient detection.
2. **Object Detection**
    - Processes a single image (chevrolet_camaro_1970.jpg) to detect objects with bounding boxes and confidence scores.
3. **Result Visualization**
    - Displays detected objects visually using the show() method.
4. **Output Saving**
    - Saves the annotated image to files/output.jpg for further analysis.

Successes

- Model loaded without errors and performed detection efficiently.
- Objects were accurately visualized and saved with annotations.

Limitations

- Processes only one image, unsuitable for video or real-time tasks.

Insights

- **Strengths:** Quick setup and testing, ideal for basic functionality verification.
- **Weaknesses:** Limited scope, no tracking or dynamic input handling.

---

2. YOLOv8 Object Detection in Video

Overview

This version extends the object detection program to process video input, enabling object detection frame by frame. The results are saved as an annotated video file, showcasing dynamic object detection.

Key Features

1. **Video Input Handling**
    - Reads a video file (ronaldo2.mp4) as input.
2. **Frame-by-Frame Object Detection**
    - Applies YOLOv8 on each video frame to detect objects.
3. **Result Visualization**
    - Annotates each frame with bounding boxes and confidence scores.
4. **Video Output Saving**
    - Saves the annotated frames into a new video file (output_video.mp4).
5. **Optional Display**
    - Displays the annotated frames during processing (can be disabled for headless environments).

Successes

- Processed video input successfully and performed real-time object detection.
- Generated an annotated video file with bounding boxes and other visual cues.
- Efficiently handled frame-by-frame processing without skipping frames.

Limitations

- No tracking capability; objects are detected per frame without continuity.
- Requires codec (mp4v) compatibility for video output.

3. Object Detection and Tracking Using YOLO and SORT

Overview

This version of the program extends object detection by integrating the SORT (Simple Online and Realtime Tracking) algorithm for tracking detected objects across video frames. It builds on the previous implementation by associating unique IDs with detected objects, enabling consistent tracking throughout the video.

Features Implemented

1. **Object Detection with YOLOv8**
    - Utilized the YOLOv8 model for detecting objects in video frames.
    - Filtered detections based on confidence scores (threshold set at 0.5).
2. **SORT Tracking**
    - Integrated the SORT tracker to assign and maintain unique IDs for detected objects.
    - Implemented a mechanism to update the tracker with new detections at each frame.
    - Tracked objects are visualized with bounding boxes and labeled with their respective IDs.
3. **Output Video Generation**
    - Saved the processed video with tracking annotations to a specified output path.
    - Displayed the annotated video frames during processing (optional).

Code Workflow

1. **Initialization**:
    - Loaded the pre-trained YOLOv8 model.
    - Initialized the SORT tracker.
2. **Detection and Tracking**:
    - For each video frame:
        - Performed object detection using YOLOv8.
        - Filtered detections based on confidence scores.
        - Updated the SORT tracker with current frame detections.
        - Drew bounding boxes and object IDs on the frame.
3. **Output**:
    - Annotated video frames were saved to a new video file.

Encountered Issue

Error Message:

```
OMP: Error #15: Initializing libiomp5md.dll, but found libiomp5md.dll already
initialized.
OMP: Hint This means that multiple copies of the OpenMP runtime have been linked
into the program. That is dangerous, since it can degrade performance or cause
incorrect results. The best thing to do is to ensure that only a single OpenMP
runtime is linked into the process, e.g. by avoiding static linking of the
OpenMP runtime in any library. As an unsafe, unsupported, undocumented
workaround you can set the environment variable KMP_DUPLICATE_LIB_OK=TRUE to
allow the program to continue to execute, but that may cause crashes or silently
produce incorrect results.
```

Root Cause:

This issue arises due to multiple instances of the OpenMP runtime library (`libiomp5md.dll`) being loaded into the program. It occurs when different libraries link their own versions of OpenMP.

Current Workarounds Attempted

1. **Environment Variable Fix**:
    - Set `KMP_DUPLICATE_LIB_OK=TRUE` to bypass the issue temporarily.
    - While this allows the program to execute, it may cause crashes or incorrect results.
2. **Avoided Static Linking**:
    - Verified library dependencies to ensure no static linking of OpenMP runtime.

Next Steps

- Identify and resolve the conflict by ensuring all libraries use a single version of the OpenMP runtime or use a different approach.

Summary

This version successfully integrates SORT tracking with YOLOv8 for object detection. However, the program encountered an OpenMP runtime conflict, which needs to be resolved for stable and accurate execution.

4. YOLOv4 Tiny with OpenCV MultiTracker

Objective:

- Test YOLOv4 Tiny object detection integrated with OpenCV MultiTracker on video files.

Code Summary:

1. **YOLOv4 Tiny Integration:** Used pre-trained YOLOv4 Tiny model with OpenCV DNN to detect objects.
2. **MultiTracker Setup:** Initialized OpenCV MultiTracker for tracking detected objects across frames.
3. **Video Processing:** Processed video frames to detect objects in the first frame or periodically, applied Non-Maximum Suppression (NMS), and tracked objects with bounding boxes.
4. **Output:** Annotated video saved to the specified output path.

Result:

- Encountered an error during execution:

  ```
  cv2.error: OpenCV(4.10.0) D:\a\opencv-python\opencv-python\opencv\modules\
  dnn\src\darknet\darknet_importer.cpp:210: error: (-212:Parsing error)
  Failed to open NetParameter file: yolov4-tiny.cfg in function
  'cv::dnn::dnn4_v20240521::readNetFromDarknet'
  ```

Conclusion:

- No attempts were made to resolve the issue at this stage.

5. Zoo Animal Tracking with YOLOv11 and Track History

Program Overview

In this program, we extended the functionality of YOLOv11 by integrating track history to visualize the movement of animals in a zoo environment. The primary aim was to create a program capable of tracking animals across video frames and plotting their trajectories.

What the Program Does

1. **Object Detection and Tracking**: The program uses a pre-trained YOLOv11 model to detect objects in each video frame and assigns unique track IDs to those objects.
2. **Track History Visualization**: It records the movement history of each detected object, retaining up to 30 frames of track data. The trajectories are displayed as lines that follow the object's movement.
3. **Video Output**: The annotated video, including bounding boxes, IDs, and trajectories, is saved to an output file.

Features

- **Detection Model**: The program uses a custom YOLOv11 model (`yolo11n.pt`) for object detection.
- **Tracking**: Object IDs are assigned to track animals across frames.
- **Trajectory Visualization**: Visualizes the movement paths of animals in the video.
- **Output Video**: Saves the processed video for further analysis or review.

Strengths

- **Custom Tracking**: Successfully tracks multiple animals and displays their movement paths.
- **Visual Output**: The program outputs clear visualizations of bounding boxes and trajectories, providing insights into animal behavior.

Limitations

1. **Object ID Changes**: The ID of an object can change in the following scenarios:
   - When the object moves rapidly.
   - When there is occlusion (e.g., objects overlapping or obstructing one another).
   - When the orientation of the animal changes significantly.
2. **Track Accuracy**: Due to ID changes, trajectories may sometimes appear discontinuous.

3. **Orientation and Occlusion Sensitivity**: The model may struggle to maintain consistent tracking when objects are partially hidden or oriented differently from their training data.

Future Improvements

- Use a more robust tracking algorithm to minimize ID switching.
- Experiment with fine-tuned YOLOv11 models trained specifically on zoo animal datasets.
- Integrate fuzzy logic to handle uncertainties caused by occlusions or fast movements.
- Increase the number of frames retained for track history to allow for longer trajectory visualization.
