DeepStream Application - Approach and Design Choices

Introduction

The DeepStream application is designed to perform object detection in videos using YOLOv5 and track individual objects across frames. The application detects and tracks objects for specified classes (e.g., person, car, truck, motorcycle, and bus) and provides a real-time object count for each class.

Approach

1. **Model Selection:** The application uses YOLOv5 as the object detection model. YOLOv5 is a state-of-the-art deep learning model that provides accurate and efficient object detection in real-time.

2. **Object Tracking:** To track objects across frames, the application uses a unique identifier (UUID) for each detected object. The identifier is stored in a dictionary along with the class label and confidence score.

3. **Output Visualization:** The application draws bounding boxes around detected objects, displays class labels, object IDs, and confidence scores on the video frames, and shows the real-time count for each class.

Design Choices

1. **Class Labels:** The application allows the user to specify the class labels for objects to detect and track. By default, the labels are set to ['person', 'car', 'truck', 'motorcycle', 'bus'], but the user can customize them as needed.

2. **Bounding Box Colors:** Different colors are assigned to each class label for better visualization. If the class label is not one of the specified colors, a default color (black) is used.

3. **Object ID Generation:** To assign a unique ID to each detected object, the application uses a UUID (Universally Unique Identifier) with 8 characters. This ensures that each object is uniquely identified.

4. **Output Video:** The application saves the processed video with bounding boxes, class labels, and object IDs along with the real-time count of each class. The output video is stored in the specified output path with the name 'output_video.mp4'.

Challenges

1. **Integration with DeepStream:** Integrating the YOLOv5 model and object tracking logic with the DeepStream pipeline required careful handling of buffers and frames to ensure smooth inference and tracking.

2. **Frame Processing Efficiency:** Processing each frame in real-time with object detection and tracking can be computationally intensive. Optimization techniques were used to ensure efficient frame processing.

3. **Class Label Mapping:** Mapping the YOLOv5 class IDs to the custom class labels for visualization and tracking required creating a dictionary to map the IDs.

Conclusion

The DeepStream application successfully detects and tracks objects in videos using YOLOv5 and provides real-time object count for each class. The chosen approach and design choices ensure the accurate and efficient processing of video data. The application can be easily extended to support additional classes or models as needed.