Truck/Ambulance Detection with YOLOv8

Approach and steps:

1. Utilized a pre-trained YOLOv8m.pt model for truck detection.
2. For ambulance detection, training will be conducted using a public dataset based on a pre-trained model.
3. Inference is using the Streamlit framework, allowing users to upload either images or videos.

Calculating Truck & Ambulance Detection Accuracy:

YOLOv8 uses Mean Average Precision (mAP) as the primary metric for accuracy evaluation, the explanation for calculate:

1. **Intersection over Union (IoU)**: This measures the overlap between predicted bounding boxes and ground truth annotations. IoU = (Area of Intersection) / (Area of Union).
2. **Precision**: The proportion of correctly detected trucks/ambulances out of all detections for that class.
3. **Recall**: The proportion of correctly detected trucks/ambulances out of all actual trucks/ambulances in the image.
4. **Average Precision (AP):** This is calculated for each class (truck and ambulance) at various IoU thresholds (typically 0.5, 0.75, and 0.95). It summarizes the trade-off between precision and recall across different IoU levels.
5. **mAP**: The mean of AP values across all classes (truck and ambulance in this case).

Calculating Inference Speed:

1. **Frames per Second (FPS):** This metric indicates how many images the model can process per second.

2. **Inference Time**: This represents the time taken for the model to process a single image. Lower inference time implies faster processing.

Improving Inference Speed:

1. **Model Selection**: Choose a smaller YOLOv8 model variant (e.g., yolov8n or yolov8s) if accuracy requirements allow. Smaller models are generally faster.
2. **Quantization**: Convert the model to a lower-precision format (e.g., half-precision) using tools like TensorFlow Lite or PyTorch Mobile. Quantization can significantly improve speed on mobile and embedded devices.
3. **TensorRT or ONNX Runtime**: Leverage hardware acceleration libraries like NVIDIA TensorRT or ONNX Runtime for optimized inference on compatible GPUs or TPUs.
4. **Optimize:** Optimize the inference code by implementing efficient algorithms to further enhance the speed of object detection.