# Technical and Conceptual Documentation for Autonomous Vehicle AI System

## Overview

The Autonomous Vehicle AI System aims to detect known objects, identify novel objects, and make collision avoidance decisions in real-time. This system is designed for rural environments, where both standard and unusual obstacles may appear. The AI framework leverages a pre-trained YOLOv8n model for known object detection, an autoencoder for detecting novel objects, and a decision-making process that handles collision avoidance based on sensor data and detected objects.

## Methodology

## 1. System Components

### 1.1. YOLOv8n for Known Object Detection

#### 1.1.1. Concept:
YOLOv8 (You Only Look Once, version 8) is a real-time object detection model that can efficiently detect and localize multiple objects in an image using a single neural network. It provides bounding boxes, class labels, and confidence scores for the objects it identifies. Fine-tuning YOLOv8n on a dataset specific to rural environments allows it to detect known objects such as pedestrians, animals, vehicles, or road signs.

#### 1.1.2 Technical Details:
- Model: YOLOv8n (lightweight version optimized for speed)
- Dataset overview for fine tuning:
The KITTI dataset is a widely used benchmark in the field of computer vision, particularly for tasks related to autonomous driving and object detection. It contains images captured from a moving vehicle and includes annotations for various objects such as cars, pedestrians, and cyclists, making it suitable for training object detection models. For this project, a subset of approximately 1,000 samples was selected from the KITTI dataset.

The model is fine-tuned using a sudataset where objects  are marked with bounding boxes and class labels. The fine-tuning code and results was saved in **Known Object Detection _ YOLOv8n.ipynb** file.

Link to subset of KITTI dataset and fine-tuning model:
https://drive.google.com/drive/folders/1gHV5K7opO3FY3ojwhoIvYBFWCrO_15Pr?usp=sharing.

- Detection Output:
    - Class ID: The type of object detected (e.g., car, person, cow).
    - Confidence: A score representing the model's certainty about the object.
    - Bounding Box: Coordinates of the object's position in the image.

### 1.1.3 Integration:

The model is loaded and used in the `AutonomousVehicleAI` class. During runtime, every frame is passed through the YOLOv8n model, which returns detected objects, their locations, and their confidence scores. These objects are then used in the decision-making process for collision avoidance.

## 1.2. Autoencoder for Novel Object Detection

### 1.2.1 Concept:

An autoencoder is an unsupervised learning model used to reconstruct input data. In this system, the autoencoder is trained to recognize and reconstruct known objects. Novel objects, which were not part of the training data, will have a higher reconstruction error since the autoencoder is unable to accurately recreate them. By setting a threshold for reconstruction error, we can flag objects that are considered novel.

### 1.2.2 Technical Details:

- Architecture: A simple convolutional autoencoder with:
    - Encoder: Reduces the image to a latent representation (compressed).
    - Decoder: Reconstructs the image from the latent space.
    - Loss Function: Mean Squared Error (MSE) between the input and the reconstructed output.
- Threshold for Novelty Detection: A manually defined threshold value that differentiates known objects (low reconstruction error) from novel objects (high reconstruction error).

### 1.2.3 Integration:

The autoencoder is used in the `detect_novel_objects` method, where each frame is passed through the network to calculate the reconstruction error. If the error exceeds the threshold, the object in the frame is flagged as novel.

## 1.3. Collision Avoidance System

### 1.3.1 Concept:

The collision avoidance system is designed to process detected objects (both known and novel) and make real-time decisions to avoid obstacles. The system combines object detection results with sensor data (e.g., distance to an object, speed) to either continue normal operation, slow down, or stop the vehicle.

### 1.3.2 Technical Details:

- Input:
    - Detected Objects: Output from YOLOv8n and the autoencoder (novel objects).
    - Sensor Data: Data such as distance to the nearest object, vehicle speed, and GPS.

- Decision Logic:
    - If any novel object is detected, the system prioritizes evasive actions.
    - If a known object is detected with a high confidence level, it evaluates the object's distance  using sensor data to determine if slowing down or stopping is necessary.
    - Rules are applied based on sensor inputs (e.g., stopping if the distance to an object is below a threshold).

### 1.3.3 Integration:

The decision-making process is handled in the `make_collision_avoidance_decision` method, where all detected objects and sensor data are combined to produce an action, such as "proceed with caution," "slow down," or "evasive action taken."

## 1.4. Class Structure: `AutonomousVehicleAI`

This class was placed in **system.ipynb** file as a main architecture. Run file **system.ipynb** to start the system.

### 1.4.1. Attributes

- `self.known_object_model`: YOLOv8n model for detecting known objects in the environment.
- `self.novel_object_model`: Autoencoder model used to detect novel objects by measuring reconstruction errors.
- `self.collision_avoidance_model`: (Optional) Placeholder for more advanced decision-making models.

### 1.4.2. Methods:

- __init__(self, yolo_model_path): Initializes the AI system by loading the fine-tuned YOLOv8n model and building the autoencoder.

- build_autoencoder(self): Builds a simple convolutional autoencoder that is used for anomaly detection (novel objects).

- detect_known_objects(self, frame): Processes a video frame using YOLOv8n to detect known objects. Returns a list of detected objects with class IDs, confidence scores, and bounding boxes.

- detect_novel_objects(self, frame): Processes a video frame using the autoencoder to detect novel objects based on reconstruction error. Returns `True` if a novel object is detected, and `False` otherwise.

- make_collision_avoidance_decision(self, detected_objects, sensor_data): Processes the detected objects (known and novel) and sensor data to make a decision about how the vehicle should respond. Actions include evasive action, slowing down, or proceeding with caution.

- process_frame(self, frame, sensor_data): Combines known object detection, novel object detection, and collision avoidance decision-making. This method processes a frame and sensor data and returns the final decision.

## 2. Data Flow

### 2.1. Input:

- Frame Data: A frame captured from the vehicle’s camera.
- Sensor Data: Distance to obstacles, vehicle speed, and other sensor information.

### 2.2. Processing:

- The frame is processed by YOLOv8n to detect known objects.
- The frame is also passed through the autoencoder to detect novel objects based on reconstruction error.
- The collision avoidance logic then combines object detection results with sensor data to determine the appropriate action.

### 2.3. Output:

The system outputs a decision, such as "proceed with caution," "slow down," or "evasive action taken."

## Conclusion

The Autonomous Vehicle AI System leverages advanced machine learning techniques to address the unique challenges of navigating rural environments. By fine-tuning the YOLOv8n model on a carefully selected subset of the KITTI dataset (subset), the system is capable of accurately detecting known objects such as pedestrians, and vehicles. In addition, the use of an autoencoder for novel object detection ensures that the system can flag and respond to unfamiliar obstacles that may not have been part of the training data and need more improvement.

The integration of known object detection, novel object identification, and collision avoidance decision-making allows the system to make informed, real-time decisions that prioritize safety and adaptability. While the current implementation is robust, there are future opportunities to enhance the system further by incorporating more advanced models for collision avoidance and novel object detection, as well as integrating additional sensor data to improve overall accuracy.

Overall, this AI system demonstrates the potential to improve safety and efficiency in autonomous vehicle operations, particularly in rural areas where unexpected obstacles are more likely to occur. With continued improvements and refinements, the system has the potential to become an essential tool for autonomous navigation in diverse environments.

## Future Improvements

- Advanced Collision Avoidance: Implement a more sophisticated collision avoidance system using reinforcement learning or imitation learning based on real-world driving data.

- Enhanced Novel Object Detection: Consider using more advanced anomaly detection models, such as variational autoencoders (VAE) or generative adversarial networks (GANs), for better generalization to unseen objects.

- Multi-Sensor Integration: Combine data from other sensors such as LiDAR, radar, or GPS to improve decision-making accuracy and robustness in complex driving conditions.