HYBRID AI CAR DRIVING SIMULATOR

A Comparative study of Rule-Based Safety, Reinforcement Learning and Machine Learning

Under the Guidance of

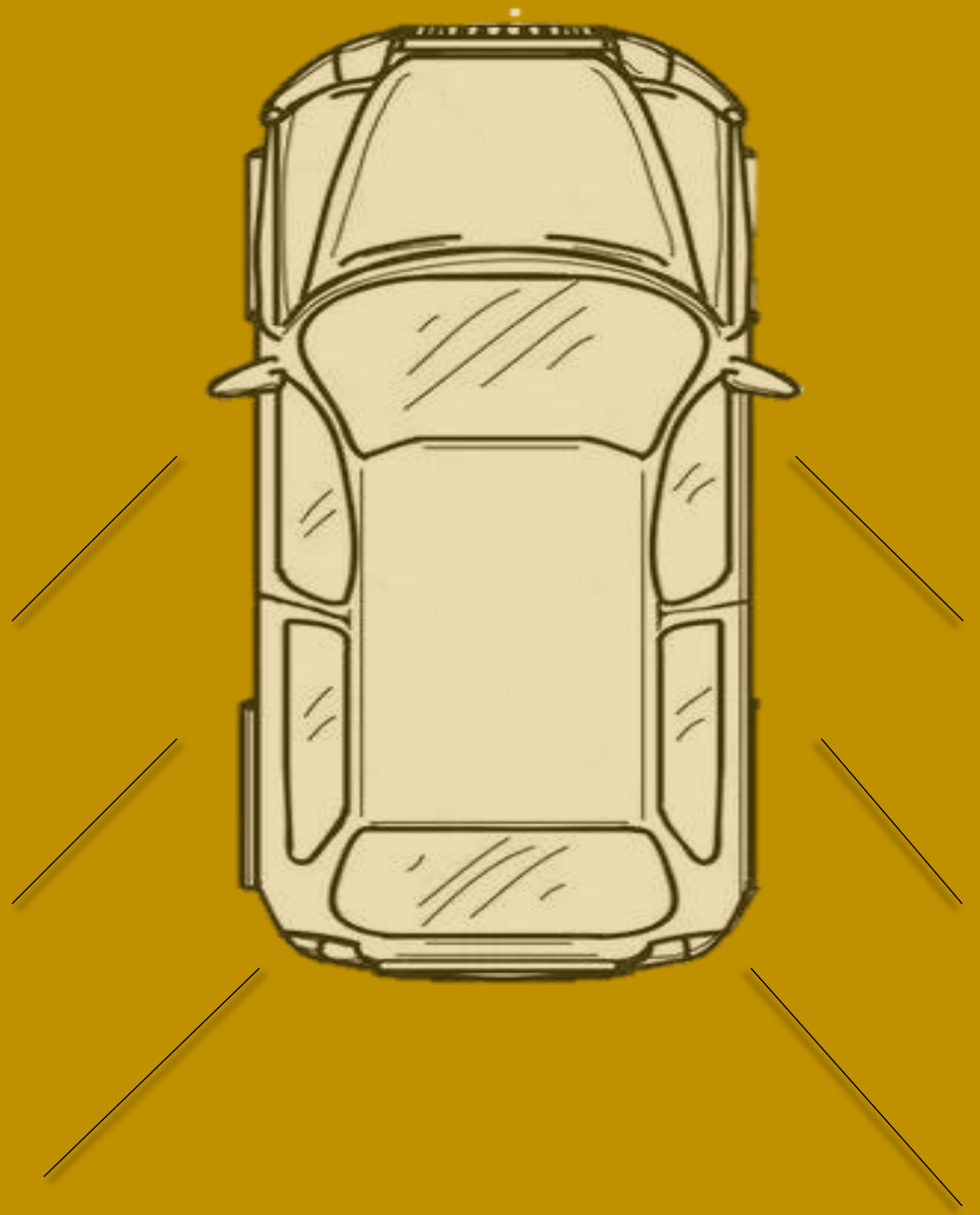
Mr. Biplab Kumar Mondal
Assistant Professor

Presented By:
Ayan Kundu
Debarshi Chaterjee
Moupiya Das
Rohit Saha

Need for Hybrid AI in Autonomous Driving

Autonomous driving is one of the most important real-world applications of Artificial Intelligence.
Machine Learning models are widely used for decision making in self-driving vehicles.

However, pure Machine Learning systems may fail in unknown or unsafe situations.
Safety is a critical requirement for autonomous driving systems.
To address this issue, this project introduces a Hybrid AI approach that combines learning and safety mechanisms.



Problem Statement

⚡ **Static Environment Limitation**

Traditional pathfinding algorithms (Dijkstra, A*) are effective in static, single-goal environments
These algorithms fail in dynamic and multi-goal scenarios due to lack of adaptability

❌ **Poor Adaptability**

These algorithms fail in dynamic and multi-goal scenarios due to lack of adaptability, requiring full re-computation for new situations, which leads to inefficiency and suboptimal decisions.

⏱️ **High Computational Cost:**

Frequent path re-computation is needed when obstacles or goals change, leading to slower performance and inefficient navigation.

⚠️ Unsafe Decisions:

Inefficient routing and inability to adapt can result in unsafe outcomes.

💡 Need for Intelligent System:

An adaptive navigation system is required for safe, efficient, and real-time decision-making.

Project Objective

1. To design and develop a 2D car driving simulation environment for autonomous driving experiments.

2. **compare the performance of two decision-making approaches:**
Hybrid AI System (Rule-based safety + Machine Learning**). Pure Machine Learning System**

3. **To ensure safe and reliable driving behavior by focusing on:**
   - **Collision avoidance**
   - **Traffic rule compliance**
   - **Smooth and controlled vehicle movement**

4. To design and develop a 2D car driving simulation environment for autonomous driving experiments.

Core Technology Used

Python

Used to build the entire simulator, AI logic, training, and evaluation system.

Py-game

Creates the 2D car driving environment and handles real-time simulation.

Random Forest Classifier

Used for driving action classification and direction control.

Q-Learning (RL)

Helps the car learn optimal driving decisions using rewards.

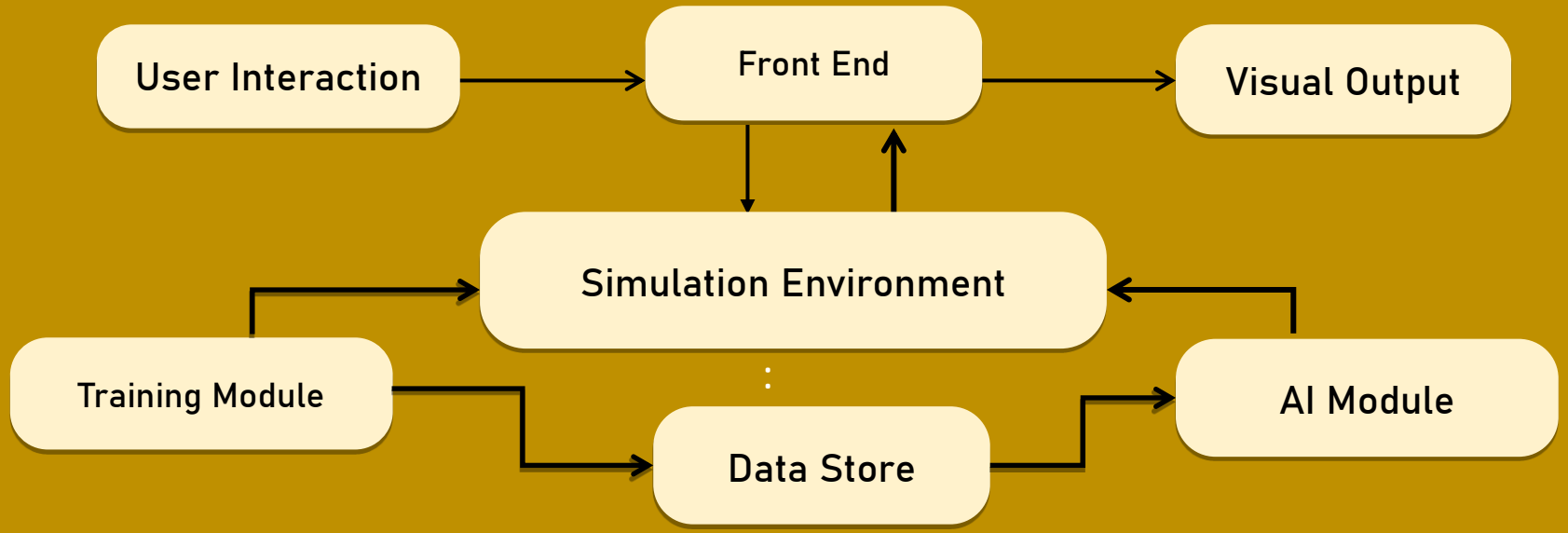
Random Forest Regressor

Predicts smooth and continuous speed control values.

Matplotlib

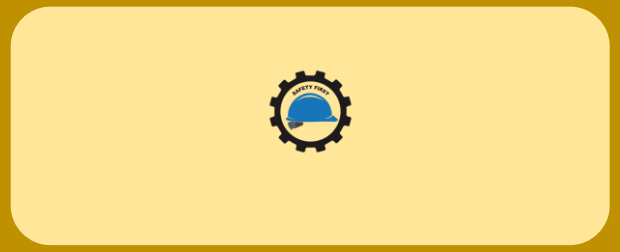
Visualizes performance results and comparison graphs.

System Architecture Overview

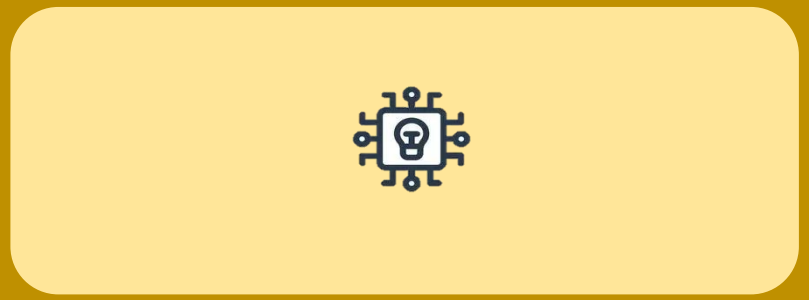


User Interaction: Keyboard inputs and GUI controls
Frontend: Py-game based graphical interface
Visual Output: Real-time simulation rendering and HUD
Simulation Environment: Road, traffic, obstacles, and sensors
AI Module: Hybrid decision-making (Rules + RL + ML)
Training Module: Data logging and model training
Data Store: CSV/JSON datasets and trained models

Hybrid AI Approach



Rule-Based Safety System
Ensures collision avoidance, lane safety, and traffic rule compliance.



Reinforcement Learning (Q-Learning)
Learns optimal driving actions based on rewards and penalties from the environment.

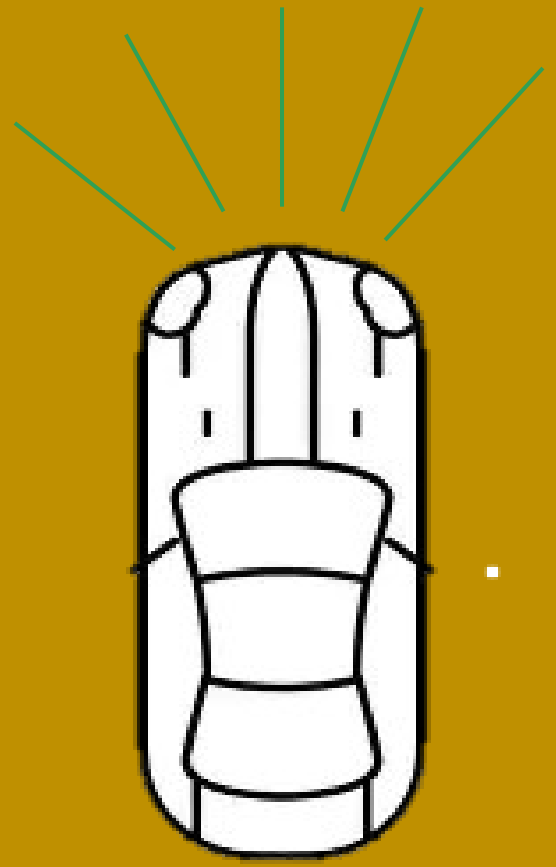
Supervised Machine Learning (Random Forest)
Predicts driving actions and speed using trained data.

This layered approach ensures that safety rules can always override ML decisions, guaranteeing robust and accident-preventing behavior

Sensor & Environment Design

The simulator uses multiple distance sensors based on ray-casting to perceive the surroundings.
These sensors detect obstacles, lane boundaries, and traffic signals in real time.

The environment is designed to simulate realistic driving conditions, including roads with lanes, obstacles, and traffic lights.
Sensor data from this environment is continuously provided to the AI system for safe and intelligent decision-making.



Training & Data Collection

1. Data Acquisition

Training data is collected during both **manual driving** and **AI-controlled driving**.

2. Parameter Logging

At each simulation step, key parameters such as **sensor distances, vehicle actions, speed, and reward values** are recorded.

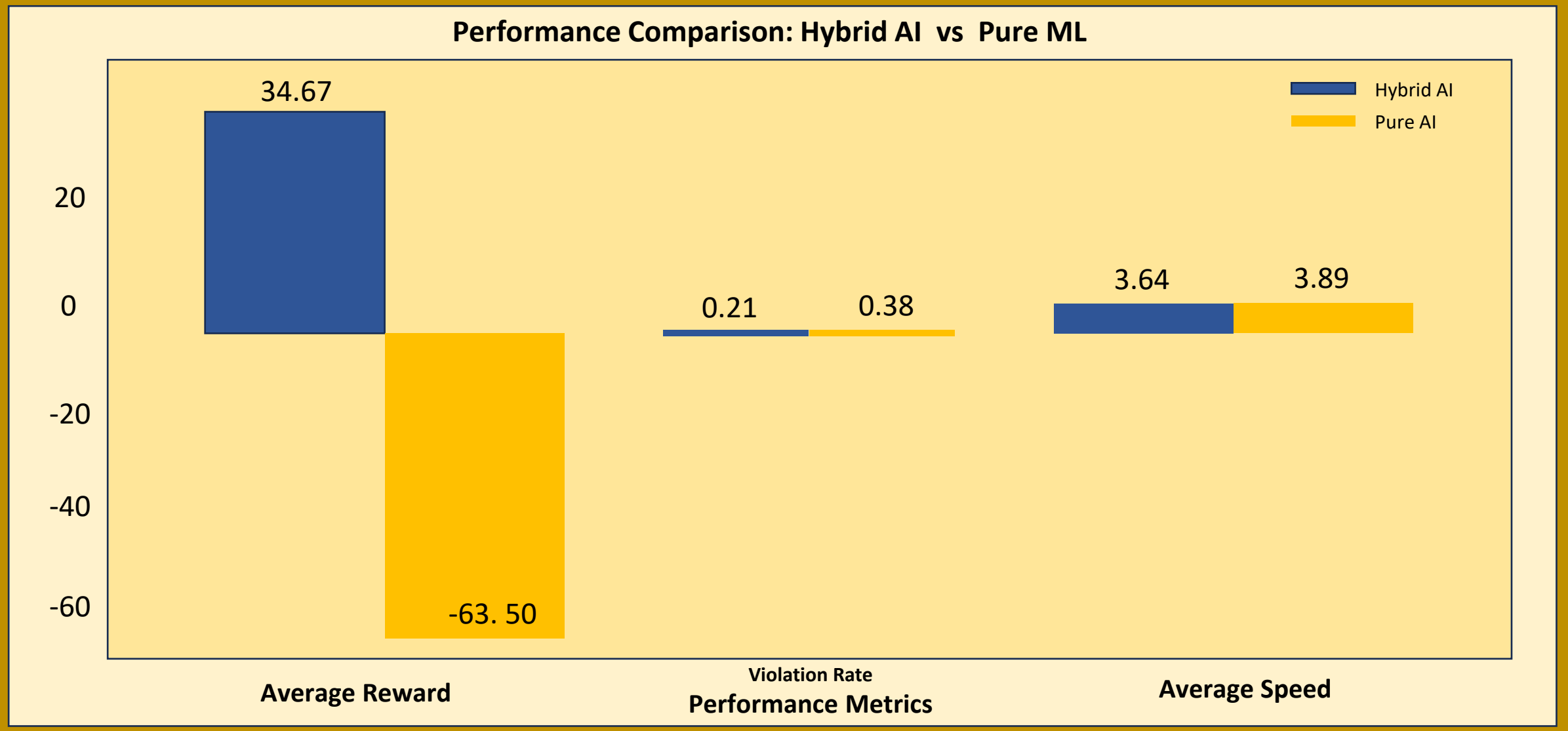
3. **Synthetic Data Generation:**

Artificial data is generated to represent **rare and risky scenarios** like collisions and traffic rule violations.

4. **Model Training Usage**

The collected data is used for **offline model training** and **online learning** to improve real-time driving performance.

Evaluation & Performance Comparison



Results & Observations

- Hybrid AI System Performance:
  The Hybrid AI system achieves a higher average reward, indicating better overall driving efficiency and goal achievement during simulation.

- Safety and Rule Compliance:
  It shows lower collision rates and fewer traffic rule violations, as safety rules override risky decisions made by learning models.

- Pure ML System Behavior:
  The Pure ML system maintains higher average speed, but often makes unsafe decisions due to the absence of explicit safety constraints.

- Overall Observation:
  These results confirm that integrating safety rules with learning algorithms improves decision stability, safety, and reliability compared to using Machine Learning alone.

Advantages of the System

- ❑ Safety rules cannot be violated preventing catastrophic failures.
- ❑ Faster and More Stable Learning. Better Generalization to Edge Cases.
- ❑ Easier Validation of important rules.
- ❑ Performance benefit due to computationally cheap rule checks
- ❑ New algorithms and sensors can be added easily.
- ❑ Modular system design makes maintenance easier.

Suitable for:

- Academic projects
- AI research
- Autonomous driving education

Limitations & Future Enhancements

- **Model Limitation:** Random forest has a lot of constraints that can be mitigated with DNN implementation
- **Limited Scalability**: CSV and JSON files that store data can easily expand exponentially and needs to be pruned
- **Limited Sensor Realism:** Sensor data has limited data point that also lacks a lot of inconvenience a real system will face
- **Reward Function Sensitivity:** Q-Learning is to reward sensitive and will be replaced by DQN for better balancing
- **Simplified Traffic Behavior**: Limited Environmental Diversity leads to low training circumstances.
- **Hard-Coded Parameters**: RL rewards, discount factors are fixed and need manual tuning. This will be replaced with self tuning systems in future.

THANK YOU