AI

Semester

Project

Presented by:

Afnan Hassan

Ali Ahmed

Minahil Ali

AI Racing Driver for TORCS

Introduction

An intelligent self-driving agent that learns to drive in the TORCS racing simulator using a multi-output neural network trained on real driving data.

Project Overview This project implements a machine-learning pipeline for building an AI-controlled racecar in the TORCS simulator. It processes human driving data, trains a neural network on key sensory features, and uses the model to predict driving actions in real-time.

Key Features

- Neural Network-Based Driving: Predicts steering (continuous), acceleration, and braking (binary).
- Real-Time Control: Integrated with TORCS via a Python socket-based client.
- Modular Pipeline: Cleanly separated data processing, training, and inference stages.

Our Approach

Initially, we explored reinforcement learning techniques to train the AI driver. We experimented with Genetic Algorithms (GA), Deep Deterministic Policy Gradient (DDPG), and NEAT (NeuroEvolution of Augmenting Topologies). Despite training different models with various architectures, the performance was suboptimal compared to supervised learning approaches. After extensive experimentation,

we concluded that the best results were achieved using simple supervised learning models.

Model Training Architecture

Input Features A total of 37 features are used, drawn from sensor readings such as:

- Track sensors: Track_1 to Track_19 (distance to track edges)
- Velocities: SpeedX, SpeedY, SpeedZ
- Car state: Angle, TrackPosition, RPM, Damage
- Wheel spin and opponent proximity These features provide spatial awareness and kinematic context.

Output Targets

- Steering: A continuous value between -1 and 1 (scaled using MinMaxScaler)
- Acceleration and Braking: Thresholded as binary actions

Network Architecture Implemented using Scikit-learn's MLPRegressor:

Layer Type Configuration Input 37 features Hidden 1 64 neurons, ReLU Hidden 2 64 neurons, ReLU Output 3 outputs: Steering, Acceleration, Braking

Additional hyperparameters:

- learning_rate='adaptive', early_stopping=True, max_iter=1000
- batch_size=64, tol=1e-5, n_iter_no_change=15

Data Handling & Training Strategy

1. CSV Aggregation: Multiple driving sessions are combined into one dataset.
2. Preprocessing:

- Missing values filled using feature-wise means
- Normalization:
    - StandardScaler for input features
    - MinMaxScaler (range -1 to 1) for steering output
    - Binary thresholds applied to acceleration and braking

3. Splitting: The dataset is divided into 60% training, 20% validation, and 20% testing.
4. Training: Uses validation split for early stopping and adaptive learning rate.

System Requirements

- Python 3.6+
- Libraries: scikit-learn, pandas, numpy, matplotlib, joblib
- TORCS + SCRC patch (socket-based control)

Conclusion

he development of the AI Racing Driver for TORCS demonstrated that while reinforcement learning methods like GA, DDPG, and NEAT were promising, supervised learning proved to be more effective for this task. The final model, built with a multi-layer perceptron architecture, effectively predicts steering, acceleration, and braking actions, offering reliable real-time control in the TORCS simulator. This outcome emphasizes the importance of model selection when building intelligent control systems for complex environments.