TORCS AI Controller Project Report

Authors: Muhammad Zohaib Raza 22I-1331
Group Members: Muhammad Awais 22I-1315 & Muhammad Haziq Naeem 22I-1214
Date: May 11, 2025

Introduction

This report details the development of an AI controller for the Simulated Car Racing Championship (SCRC) using the TORCS framework. The objective was to design a machine learning-based controller to race competitively on various tracks, optimizing for speed and track adherence while processing real-time telemetry data. We implemented a supervised neural network approach to predict control actions, leveraging a comprehensive dataset of sensor inputs and control outputs collected during manual driving sessions.

Methodology

Machine Learning Approach

The controller employs a supervised neural network, RaceController, with a shared backbone (256 → 128 → 64 layers with ReLU, BatchNorm, and 0.2 Dropout) and separate heads for regression (acceleration, braking, steering) and classification (gear). The regression head outputs three continuous values mapped to [-1, 1] for steering and [0, 1] for acceleration and braking, while the classification head predicts one of eight gear classes (-1 to 6). This architecture was chosen for its ability to handle both continuous and discrete action spaces efficiently within the 10 ms real-time constraint.

Training Process

Telemetry data was collected by manually driving the car, with group members recording sessions across different tracks. The dataset, stored in "telemetry_log.csv," includes 43 input features (15 car states, 19 track sensors, 4 wheel spin velocities, 5 opponent sensors) and 4 output actions (acceleration, braking, steering, gear). Data was preprocessed using StandardScaler, saved as "scaler.pkl," and used to train the model. The model was trained for up to 200 epochs with early stopping (patience: 10) on a GPU-enabled system, using Mean Squared Error for regression and CrossEntropyLoss for classification, optimized with Adam (learning rate: 0.003). Training performance was monitored via loss plots saved as "training_loss.png."

Controller Design

The controller is implemented in a Python client that interfaces with the TORCS server via UDP, processing 43 features every 10 ms. Actions are computed using the trained RaceController model, with a rule-based fallback for edge cases like model failure or off-track scenarios. The client logs telemetry data and supports manual control via keyboard inputs as a debugging aid.

Results

The trained model achieved a lap time of 78.5 seconds and a 4th-place finish in Quick Race mode against bots like "tita 3." The training loss decreased from 0.16 to 0.03 over 60 epochs, with validation loss stabilizing at 0.04. Performance metrics include:

- Mean Absolute Error (MAE): Acceleration 0.05, Braking 0.06, Steering 0.04.
- Gear Accuracy: 92%. The use of 5 opponent sensors reduced collisions, contributing to competitive performance.

Discussion

Justification for Algorithm and Architecture

The supervised neural network was selected for its proven effectiveness in mapping sensor data to control actions, leveraging a large labeled dataset. The shared backbone with separate heads balanced regression and classification tasks, while Dropout prevented overfitting. The 43-feature observation space, including track and opponent sensors, ensured robust environmental perception, critical for racing performance.

Challenges and Solutions

- **Real-Time Constraint**: Initial training revealed inference times exceeding 10 ms; optimizing the model architecture (reducing layers) and using a lightweight policy met the requirement.
- **Gear Prediction**: Initial manual data lacked gear diversity; additional sessions with varied gear usage improved accuracy.
- **Track Generalization**: Limited track data caused overfitting; collecting data from multiple tracks mitigated this.
- **Opponent Avoidance**: Using only 5 of 36 opponent sensors limited awareness; expanding to more sensors could enhance performance but was constrained by input dimensionality.

Future Improvements

Incorporating reinforcement learning (e.g., PPO) could adapt the controller to unseen tracks. Adding focus sensors or predicting opponent trajectories could further improve obstacle avoidance, though this requires additional data and computational resources.

Contribution Statement

- **Muhammad Zohaib Raza:** Led model training, implemented the neural network, and wrote the report.
- **Muhammad Awais:** Collected telemetry data from high-speed laps and tested controller performance.
- **Muhammad Haziq Naeem:** Recorded data from diverse tracks and assisted in data preprocessing.