TORCS AI Driver Training and Selection System Report

Overview

This report outlines the approach taken to design, train, and select track-specific AI controllers for The Open Racing Car Simulator (TORCS). The system implements a flexible architecture allowing users to train multiple specialized models and select the appropriate model at runtime based on track characteristics.

System Architecture

Our system consists of three key components:

1. **Training System**: A data-driven approach to learn optimal driving behaviors from track-specific datasets
2. **Model Selection System**: A runtime interface allowing users to select the appropriate model for each track
3. **AI Controller**: The neural network-based controller that processes sensor inputs and generates control outputs

Training System: Data-Driven Approach

The training system employs supervised machine learning to model driving behavior:

Data Collection and Pre-processing

- Track-specific driving data is collected in CSV format from expert drivers or previous simulations
- Each dataset contains:
    - **Input features**: Track sensors, car state (speed, angle, RPM, trackPos, etc.)
    - **Output targets**: Control actions (steering, acceleration, braking, gear, clutch)
- Data preprocessing includes:
    - Handling missing values and normalization using StandardScaler

    - Creation of consistent feature mappings between dataset columns and standardized names

Model Architecture

The controller is implemented as a Multi-Layer Perceptron (MLP) neural network with:

- Three hidden layers (512, 256, 128 neurons) with ReLU activation
- Adam optimizer with adaptive learning rate
- L2 regularization to prevent overfitting
- Early stopping to preserve generalization capabilities

This architecture provides a good balance between:

- Learning complex, non-linear driving behaviors
- Maintaining computational efficiency for real-time prediction
- Generalizing effectively to similar but unseen track conditions

Track-Specific Training

A key innovation is our track-specific training approach:

1. The enhanced `train_torcs_ai.py` allows users to select from available datasets at training time
2. Model parameters and metadata are saved in track-specific directories (e.g., `track1_model`, `speedway_model`)
3. Each track model is optimized for the particular characteristics of that track

Model Selection System

To leverage the track-specific models during gameplay, we implemented a model selection system that:

1. Automatically discovers available model directories at runtime
2. Allows selection through multiple interfaces:
    - Command-line arguments (`--model track1_model`)
    - Environment variables (`TORCS_MODEL_PATH=track1_model`)
    - Interactive terminal prompt with numbered list of available models

This approach provides flexibility for different usage scenarios:

- Scripted automation through command-line flags
- System-wide configuration via environment variables
- Interactive selection for exploratory gameplay

AI Controller Implementation

The `TORCSAIDriver` class implements the core driving intelligence:

Feature Extraction and Mapping

The controller dynamically maps between the standardized TORCS state representation and the dataset-specific feature names:

- Dynamically extracts track sensors, wheel velocities, and car state
- Maps between standardized names (e.g., 'angle') and dataset-specific column names (e.g., 'Angle')

Neural Network Prediction

The model prediction process:

1. Extracts and normalizes features from the car state
2. Passes normalized features through the neural network
3. Interprets outputs as control actions (steering, acceleration, braking, gear, clutch)
4. Applies post-processing to ensure valid control range

Advanced Control Logic

Beyond simple neural network prediction, several advanced control mechanisms ensure robust performance:

1. **Control Smoothing**: Prevents jarring transitions between control outputs
2. **Gear Management**: Advanced RPM-based gear selection with fallback to speed-based shifting
3. **Stuck Detection**: Identifies when the car is not moving and applies recovery maneuvers
4. **Safety Constraints**: Ensures acceleration and braking aren't applied simultaneously

Training and Evaluation Process

Our training process involves:

1. **Data Loading**: Automatically identifying and validating CSV data format
2. **Feature Selection**: Identifying the core control features and sensor inputs
3. **Model Training**: Optimizing neural network weights using the training dataset
4. **Evaluation**: Measuring performance using MSE, R^2 score, and visual prediction analysis
5. **Export**: Generating a self-contained driver model with all necessary metadata

Conclusion

The described approach offers several advantages for TORCS AI controller development:

1. **Specialization**: Track-specific models can capture the nuances of different racing environments
2. **Flexibility**: Runtime model selection allows easy adaptation to different tracks
3. **Robustness**: Combination of learned behavior and safety constraints ensures reliable control
4. **Extensibility**: New track models can be added without modifying the core controller code

This architecture provides a solid foundation for further refinements, such as:

- Ensemble models combining multiple track-specific controllers
- Integration of reinforcement learning for policy improvement
- Implementation of dynamic model switching based on track section detection

The system demonstrates how supervised learning can be effectively applied to the complex domain of simulated racing while maintaining the flexibility needed for practical gameplay scenarios.