**Artificial Intelligence (AI 2002)**

**Final Project Report**

**Group members:**

Shayane Zainab, 22i-1049, Section J

Zainab Fatima, 22i-1064, Section J

Haris Ahmed, 22i-1124, Section J

**1. Objective**

The goal of this project is to train a machine learning-based controller that can drive a simulated vehicle in TORCS (The Open Racing Car Simulator) by predicting throttle, brake, and steering commands based on real-time sensor data. We use a data-driven approach where a neural network model is trained on human-driven telemetry data to learn control mappings.

**2. Data Collection Process**

Telemetry data was collected during manual driving sessions in TORCS using the pyclient.py script. This script listens to real-time sensor data such as:

* speedX, speedY, speedZ: representing the car's velocity in different directions.
* rpm: engine revolutions per minute.
* gear: current transmission gear.
* trackPos: lateral position of the car on the track.
* angle: angle between the car's heading and the track axis.

At the same time, driver inputs were recorded, including:

* accelerate: throttle (0.0 to 1.0)
* brake: braking force (0.0 to 1.0)
* steer: steering angle (−1.0 to 1.0)

The collected data was stored in a CSV file telemetry\_log.csv, with each row representing a timestamped instance of sensor readings and corresponding control inputs.

**3. Data Preprocessing**

The training script (train\_model.py) performs the following preprocessing steps:

* **Cleaning**: All rows with missing (NaN) values were removed using df.dropna() to ensure data integrity.
* **Feature Selection**: Seven input features were selected:  
  ['speedX', 'speedY', 'speedZ', 'rpm', 'gear', 'trackPos', 'angle']  
  These features were chosen due to their strong correlation with vehicle dynamics and control.
* **Normalization**: All input features were normalized using StandardScaler to transform them into zero-mean, unit-variance values. This ensures faster and more stable training.
* **Train-Test Split**: The dataset was split into training and testing sets using train\_test\_split with a 90-10 ratio, to assess model generalization.

**4. Model Architecture and Training**

We used **Scikit-learn’s MLPRegressor** to implement a fully connected feedforward neural network with the following configuration:

* **Hidden Layers**: Two hidden layers, each with 64 neurons
* **Activation Function**: ReLU (rectified linear unit)
* **Optimizer**: Adam
* **Max Iterations**: 300 epochs

The model is trained to map sensor inputs to three outputs:

* accelerate
* brake
* steer

The output is a continuous-valued vector suitable for control signals.

Post-training, the model and scaler were saved using joblib to torcs\_mlp\_model.pkl and input\_scaler.pkl.

**5. Autonomous Controller Integration**

In autoDrive.py, the trained model is loaded and integrated into the AutonomousDriver class. The workflow is:

* **Sensor Parsing**: Incoming string data from TORCS is parsed into key-value sensor pairs.
* **Input Formatting**: Input values are extracted in the same order as during training and reshaped into a NumPy array.
* **Scaling**: Inputs are scaled using the previously fitted StandardScaler.
* **Prediction**: The MLP model predicts continuous values for acceleration, braking, and steering.
* **Auto Gear Control**: A rule-based function determines the appropriate gear based on speed and RPM, ensuring smooth gear transitions.

The final control command is formatted as:

text

CopyEdit

(accel X) (brake Y) (steer Z) (gear N)

and sent back to the TORCS simulator.

**6. Summary**

This approach combines supervised learning with real-time sensor integration. The MLP model effectively generalizes human driving behavior and offers consistent performance when driving autonomously in TORCS. Future improvements can include:

* Collecting more diverse driving data (e.g., on different tracks, weather conditions).
* Using deep reinforcement learning to adaptively learn from environment feedback.
* Predicting gear using the model instead of rule-based logic.