# Report: Decision Making for Autonomous Vehicles on Highways --Group 41

## Github: https://github.com/yYang-bit/ME5418.git

## Introduction:

This project implements an autonomous vehicle lane-changing decision-making system using Deep Reinforcement Learning. The primary objective is to train an agent that can make optimal lane-change decisions in a simulated traffic environment. The Advantage Actor-Critic algorithm is employed to achieve this goal, leveraging the SUMO simulation environment for realistic interactions.

## Overview of the Neural Network

| Parameters | Definition | A2C |
|---|---|---|
| state_size | Size of the input state (observation space dimension) | 5 |
| action_size | Number of possible actions (action space size) | 5 |
| gamma | Discount factor for future rewards | 0.99 |
| epsilon | Initial exploration rate | 0.1 |
| epsilon_decay | Exploration rate decay factor | - |
| memory_size | Maximum size of the experience replay buffer | 10000 |
| Train_interval | Number of samples per training batch | 5 |
| lr | Learning rate for optimizer | 0.001 |

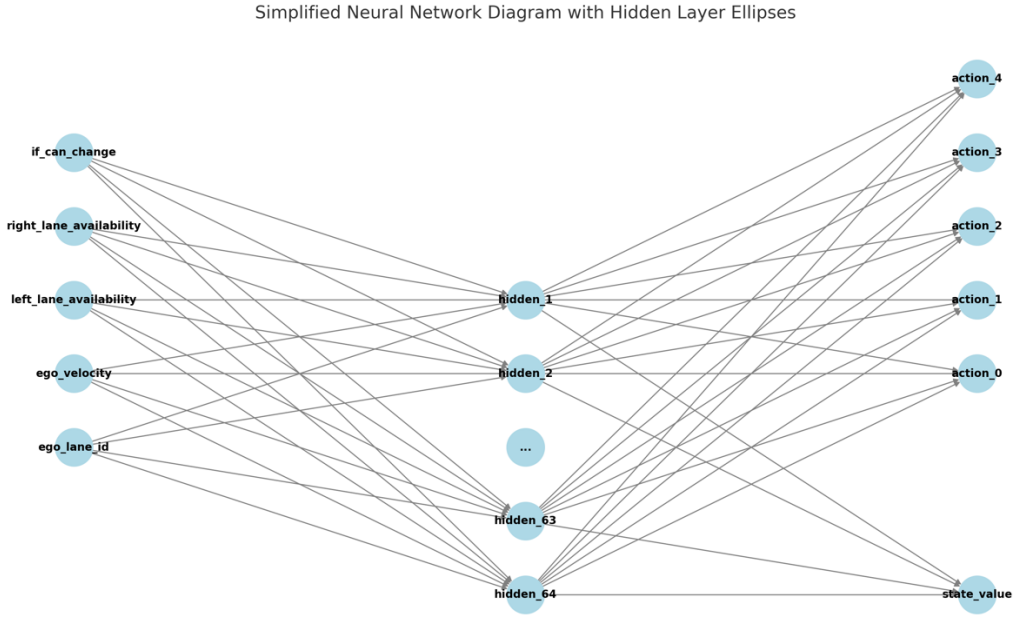



Figure 1: Neural Network Model

**Generalize before input**

Ego Lane ID: Normalized as (ego_lane_id + 1) / 4 to map lane positions between 0 and 1, helping the network generalize across different lane configurations.

Ego Velocity: Scaled by dividing by self.maximum_speed, keeping values between 0 and 1 for consistency regardless of speed range.

Left and Right Lane Availability: Binary indicators (1 or 0) showing if lane changes to the left or right are possible, guiding the agent's awareness of maneuvering space.

Can Change Lane: A binary indicator (1 or 0) signaling if a vehicle in the same lane necessitates a lane change, adding urgency to decision-making.

**Input and Output**

Input (State): A binary vector representing ego_lane_id, ego_velocity, left_lane_availability, right_lane_availability, and if_can_change (indicating if a lane change is needed). These features help the agent assess its situation within the traffic environment.

Outputs: Actor: Returns action probabilities guiding lane-change decisions. There are five possible actions: 0 (change to the left lane), 1 (change to the right lane), 2 (idle), 3 (accelerate), and 4 (decelerate). Critic: Provides a state value estimate for computing advantages during training.

The actor's output helps the agent select optimal actions, while the critic's output aids in evaluating state values to guide policy updates, enhancing learning efficiency.

## Agent-Environment Interaction

### 3.1 Observation Phase

At each step, the agent receives information about its current state, which includes ego_lane_id, ego_velocity, left_lane_availability, right_lane_availability, and if_can_change (whether there is a car ahead requiring a lane change). This information helps the agent understand the traffic situation.

### 3.2 Action Selection

The agent passes the state through the ActorCritic neural network, which gives two outputs: (1) the actor's action probabilities for five actions—0 (change to the left lane), 1 (change to the right lane), 2 (stay in the current lane), 3 (accelerate), and 4 (decelerate), and (2) the critic's estimate of how good the current state is. The agent uses these probabilities to pick an action, sometimes choosing a random action to explore new strategies (with a probability of epsilon).

### 3.3 Execution and Feedback

The chosen action is sent to the environment with env.step(action), which returns next_state, reward, and done. The reward shows how good the action was (e.g., safe driving gets positive rewards, risky behavior gets penalties). The done flag tells the agent if the episode is over (e.g., due to a collision or reaching the end of the simulation).

### 3.4 Data Collection for Training

The agent records each experience (state, action, reward, next_state, done) for training later. This data helps the agent learn and improve its policy.

We collect 5 sets of data at a time for training, and the following is the process of a certain period of time

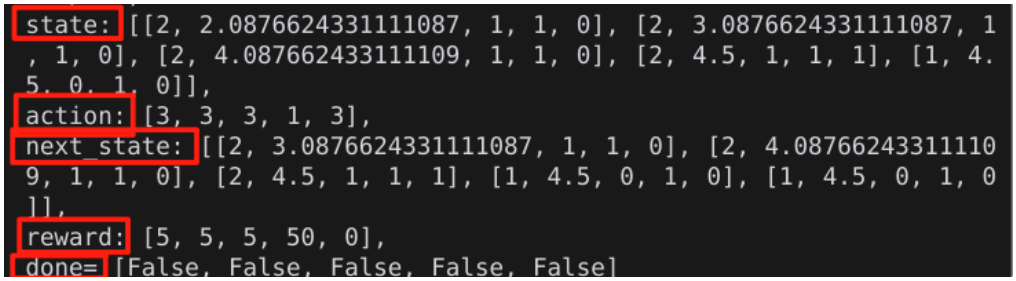



Figure 2: 5 sets of data at a time

### 3.5 Training Process

The agent trains after collecting enough data. The actor loss adjusts the agent's action choices based on how good they were (the advantage). The critic loss reduces errors in how the agent estimates state values. The training updates the network using backpropagation and a learning rate (lr) of 0.001, helping the agent learn from past experiences and make better decisions in the future.

### 3.6 How to Backpropagation in This Project

- Forward Pass: The current state is passed through the ActorCritic model to get action probabilities (actor) and state value (critic).
- Loss Computation: The actor and critic losses are calculated based on the rewards, chosen actions, and estimated state values.
- Backward Pass: loss.backward() calculates how each weight in the model should change to reduce the loss.
- Weight Update: self.optimizer.step() applies these changes to the model's weights, refining the network's predictions and decisions over time.

## Training Results Analysis

During the first training, the algorithm did not converge, so we adjusted the reward structure, including but not limited to increasing the reward for successful lane change, reducing the reward for acceleration, and reducing the penalty for unnecessary lane change.

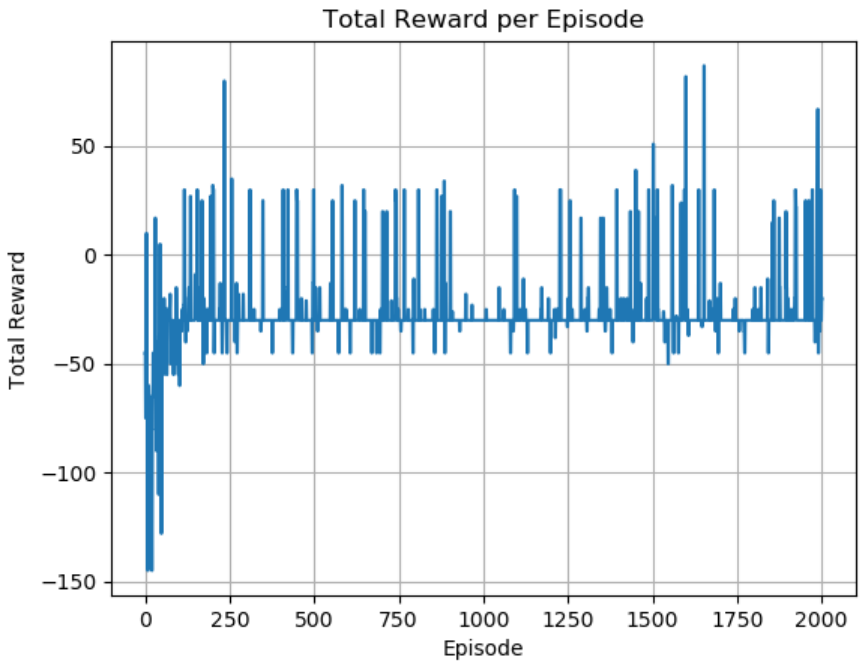



Figure 3: First time training

After many adjustments, the algorithm converges successfully. The training results for the lane-changing decision-making agent using the A2C algorithm show significant progress:

After around 600 episodes, the agent shows a steady increase in total rewards, indicating effective learning and policy development for safe and efficient lane changes. From episode 1200 onward, the agent's performance stabilizes with fewer reward fluctuations, showcasing a reliable policy capable of handling different traffic scenarios.

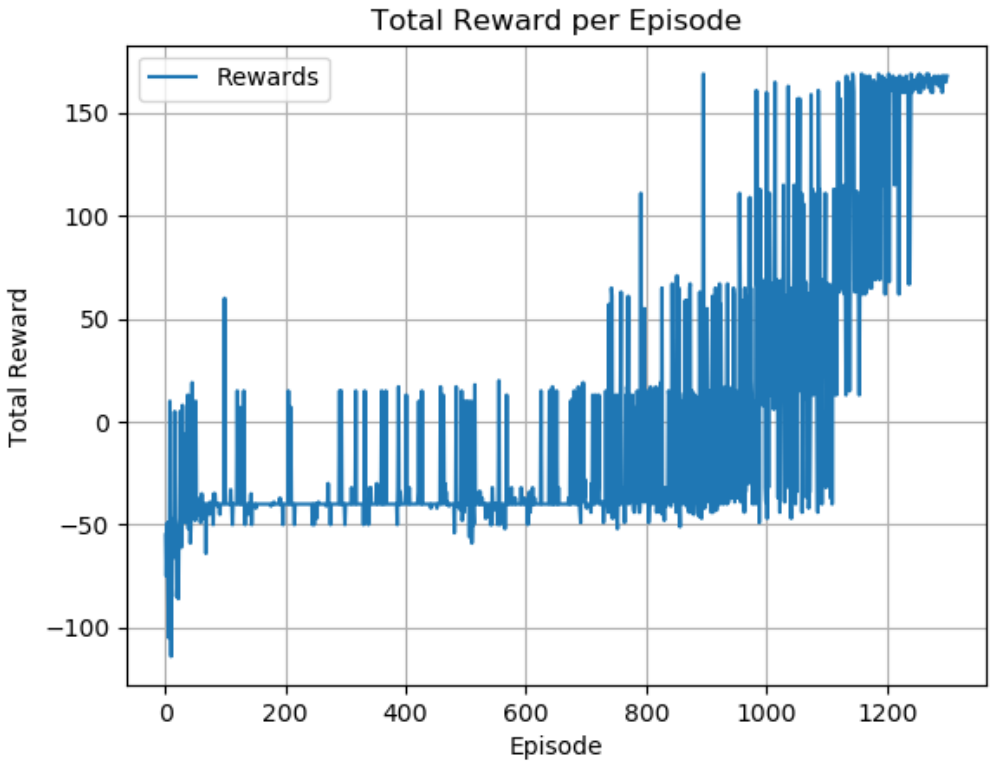
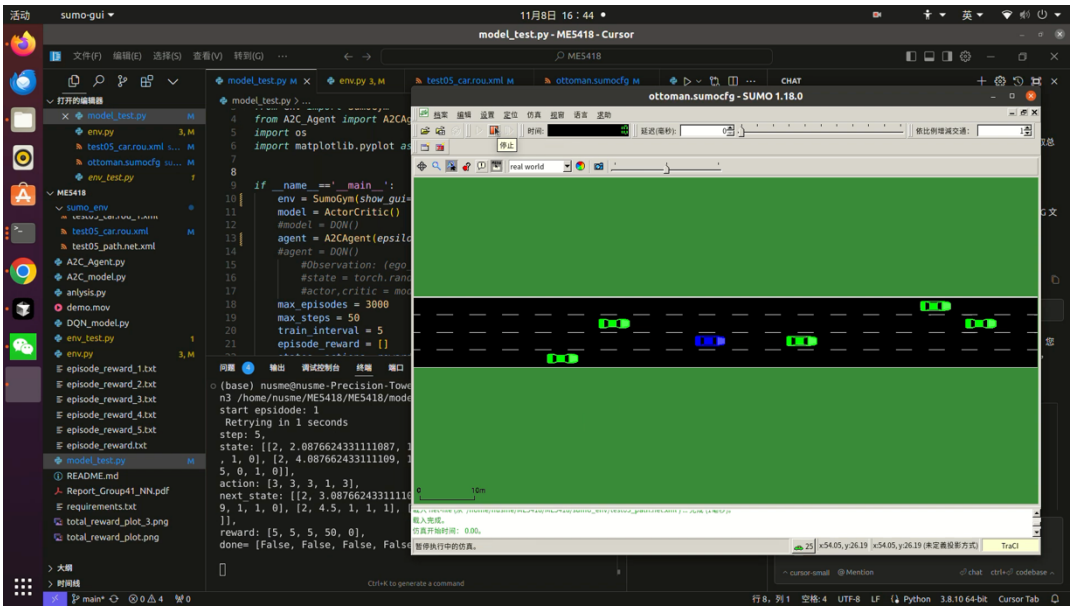



Figure 4: Training result after 1300 episodes after adjustments

## Simulation in SUMO




The video is in a zip package

## Future work

To improve the agent's performance and address current challenges, the following steps will be taken:

**Reward Structure Adjustment:** The current model shows that the agent struggles to autonomously choose the "idle" action after reaching maximum speed, resulting in less stable driving behavior.

**Increased Training Episodes:** Additional training episodes will be introduced to provide the agent with more opportunities to learn and refine its decision-making, leading to better generalization and handling of complex traffic scenarios.

**Improving Obstacle Avoidance:** Observations from recorded videos reveal that the agent sometimes fails to avoid collisions with other vehicles.