Reinforcement Learning Exercise 2

Joshua Glaspey

EEL6938: AI for Autonomous Systems

Due Date: April 21, 2025

# Table of Contents

Contents

[Table of Contents 2](#_Toc195481974)

[Introduction 3](#_Toc195481975)

[Code Modifications (Kept from Reinforcement Learning Exercise 1) 3](#_Toc195481976)

[Models 3](#_Toc195481977)

[Algorithm Hyperparameters 3](#_Toc195481978)

[Tested Hyperparameters: 3](#_Toc195481979)

[Environment Changes 4](#_Toc195481980)

[Reward Structure 6](#_Toc195481981)

[Performance Metrics 7](#_Toc195481982)

[Training Procedure 8](#_Toc195481983)

[Running the Training Script 9](#_Toc195481984)

[Results 10](#_Toc195481985)

[Algorithm Ablation Study 10](#_Toc195481986)

[Learning Rate Ablation Study 14](#_Toc195481987)

[Batch Size Ablation Study 18](#_Toc195481988)

[Buffer Size Ablation Study 22](#_Toc195481989)

[Tau Ablation Study 26](#_Toc195481990)

[Gamma Ablation Study 30](#_Toc195481991)

[Entropy Coefficient Ablation Study 34](#_Toc195481992)

[Network Architecture Ablation Study 38](#_Toc195481993)

[Chunk Size Ablation Study 42](#_Toc195481994)

[Best Configurable Results Ablation Study 46](#_Toc195481995)

[Discussion 50](#_Toc195481996)

# Introduction

In this assignment, the goal was to train an RL agent to perform Adaptive Cruise Control (ACC) by not only tracking a reference speed profile, but also maintaining a safe distance from a lead vehicle. Using a pre-existing speed-following codebase (provided by Reinforcement Learning Assignment 1), I implemented modifications to the environment, adjusted the reward structure, and added distance constraints to simulate realistic driving conditions. Additionally, I experimented with different RL models and hyperparameters to evaluate their impact on performance. This report summarizes the changes made to the system, the procedures used during training, and the results obtained through both visual and quantitative analysis.

# Code Modifications (Kept from Reinforcement Learning Exercise 1)

## Models

Like Reinforcement Learning Exercise 1, four different algorithms were implemented to evaluate their effectiveness. The default algorithm used was Soft Actor Critic (SAC), but this code implements Proximal Policy Optimization (PPO), Twin Delayed DDPG (TD3), and Deep Deterministic Policy Gradient (DDPG). Each model has its own respective configuration of hyperparameters, but for consistency, no additional hyperparameters outside the scope of what is changeable in SAC was implemented. The code below shows the implementation of each algorithm:

model = PPO(policy="MlpPolicy", env=train\_env, verbose=1, policy\_kwargs=policy\_kwargs, learning\_rate=args.learning\_rate, batch\_size=args.batch\_size, gamma=args.gamma, ent\_coef=ent\_coef\_param, device=device)

model = TD3(policy="MlpPolicy", env=train\_env, verbose=1, policy\_kwargs=policy\_kwargs, learning\_rate=args.learning\_rate, batch\_size=args.batch\_size, buffer\_size=args.buffer\_size, tau=args.tau, gamma=args.gamma, device=device)

model = DDPG(policy="MlpPolicy", env=train\_env, verbose=1, policy\_kwargs=policy\_kwargs, learning\_rate=args.learning\_rate, batch\_size=args.batch\_size, buffer\_size=args.buffer\_size, tau=args.tau, gamma=args.gamma, device=device)

## Algorithm Hyperparameters

Each of the models has a set of configurable hyperparameters. Below is the list of modifiable hyperparameters for each model.

1. SAC: Learning rate, Batch size, Buffer size, Tau, Gamma, Entropy Coefficient, Net Arch.
2. PPO: Learning rate, Batch size, Gamma, Entropy Coefficient, Net Arch.
3. TD3: Learning rate, Batch size, Buffer size, Tau, Gamma, Net Arch.
4. DDPG: Learning rate, Batch size, Buffer size, Tau, Gamma, Net Arch.

## Tested Hyperparameters:

In this experiment, the SAC algorithm was evaluated across multiple hyperparameter configurations to identify the optimal setup for model performance. The testing procedure began with a default set of hyperparameters (**bolded** in the list below), establishing a baseline for comparison. Following this initial phase, ablation studies were conducted to systematically assess the impact of individual hyperparameters. During these studies, all hyperparameters except the one being tested were held constant at their default values, while the target hyperparameter was varied across the specified range (listed below). This approach ensured a clear understanding of each parameter’s influence on model performance.

* Learning Rates: 1e-5, 1e-4, **3e-4**, 7e-4, 3e-3
* Batch Sizes: 32, 64, 128, **256**, 512
* Buffer Sizes: 50000, 100000, **200000**, 500000, 1000000
* Tau: 0.0001, 0.001, **0.005**, 0.01, 0.02
* Gamma: 0.90, 0.95, **0.99**, 0.999, 1.0
* Entropy Coefficients: **‘auto’**, 0.01, 0.05, 0.1
* Net Archs: [64x64], [128x128], **[256x256]**, [512x512]
* Chunk Sizes = 2, 10, 50, **100**, 200, 400, 600

Each hyperparameter range was carefully selected to explore different trade-offs in model performance. This list remains mostly the same as the list provided in Reinforcement Learning Exercise 1, with slight adjustments to balance learning rate and chunk sizes:

1. **Learning Rate (1e-5 to 3e-3)** – The values span several orders of magnitude to compare slow convergence with rapid learning. This broad range helps identify the optimal balance between learning speed and stability.
2. **Batch Size (32 to 512)** – The batch sizes are selected in powers of 2 to examine the effect of variance in updates. Smaller batches introduce more variability, which can enhance exploration, while larger batches provide more stable but computationally expensive updates.
3. **Buffer Size (50,000 to 1,000,000)** – This range tests the impact of memory capacity on training. Smaller buffers prioritize recent experiences, while larger buffers retain extensive history at the cost of increased resource consumption.
4. **Tau (0.0001 to 0.02)** – This parameter controls the rate of network updates. Lower values ensure smoother, incremental updates, while higher values enable faster adaptation at the risk of increased variability.
5. **Gamma (0.9 to 1.0)** – The discount factor determines the weighting of future rewards. Lower values emphasize short-term gains, whereas higher values favor long-term rewards, influencing overall policy behavior.
6. **Entropy Coefficient (‘auto’ / 0.0, 0.01 to 0.2)** – This regulates the exploration-exploitation balance. Higher values encourage more exploration, while the ‘auto’ setting dynamically adjusts based on policy confidence. For the SAC model, the default value is ‘auto’ and for the PPO model, the default value is 0.0.
7. **Network Architecture ([64x64] to [512x512])** – The varying sizes of the neural network test the trade-off between computational efficiency and representational power. Smaller architectures train faster, while larger networks capture more complex patterns.
8. **Chunk Size (2 to 600)** – This represents the size of the split of the dataset into episodes. Specifically, the chunk size is related to the number of allowable steps within an episode. Therefore, if the chunk size is 2, the 1200 length episode is split into 600 episodes of length 2. Shorter episodes should provide more frequent updates but overall higher variance, where longer episodes should improve stability but require more training time.

## Environment Changes

The original environment was extended to simulate the ACC system by adding a lead vehicle and enforcing realistic driving constraints. The key modifications are highlighted in this section.

**1. Lead Vehicle Implementation**

The lead vehicle’s speed was derived from the reference speed profile with added noise, and its position was calculated via cumulative sum.

LEAD\_SPEEDS = FULL\_SPEED\_DATA + 1 \* np.sin(0.05 \* np.arange(DATA\_LEN)) + 0.5 \* np.random.randn(DATA\_LEN)

LEAD\_POSITIONS = np.cumsum(LEAD\_SPEEDS)

Both training and testing environments were updated to split the lead vehicle positions into episodes using the function below. This is the same methodology used to implement the existing function.

def chunk\_positions\_into\_episodes(positions\_data, chunk\_size):

    positions = []

    start = 0

    while(start < len(positions\_data)):

        end = start + chunk\_size

        position\_chunk = positions\_data[start:end]

        positions.append(position\_chunk)

        start = end

    return positions

**2. Observation and Action Space**

For each environment to use consistent “safe” values for the minimum/maximum distances and maximum acceleration/deceleration, several constants were defined as shown below.

MIN\_DISTANCE = 5.0

MAX\_DISTANCE = 30.0

MAX\_SAFE\_ACCELERATION = 2.0

MAX\_SAFE\_DECELERATION = -2.0

Therefore, the action space was constrained to these realistic acceleration and deceleration limits:

self.action\_space = spaces.Box(low=MAX\_SAFE\_DECELERATION, high=MAX\_SAFE\_ACCELERATION, shape=(1,), dtype=np.float32)

Additionally, the observation space was expanded to include the distance to the lead vehicle. The original environment only observed: . The new version adds distance to the lead vehicle as a third observation:

self.observation\_space = spaces.Box(

low=np.array([0.0, 0.0, 0.0], dtype=np.float32),

high=np.array([50.0, 50.0, 100.0], dtype=np.float32),

dtype=np.float32

)

**3. Step Function Logic**

For each timestep, the environment follows this logic shown below. This follows a successful execution of the function in both training and testing environments.

1. Force the acceleration into realistic acceleration limits (-2.0 to 2.0).

accel = np.clip(action[0], MAX\_SAFE\_DECELERATION, MAX\_SAFE\_ACCELERATION)

1. Use the current and previous acceleration to calculate jerk.
   1. Jerk = The difference between the current and previous accelerations divided by the change in time between time steps.

jerk = (accel - self.prev\_accel) / self.delta\_t

1. Calculate the current ego vehicle’s speed using acceleration and time.
   1. Make sure it is always non-negative.

self.current\_speed += accel \* self.delta\_t

if self.current\_speed < 0:

self.current\_speed = 0.0

1. Update the ego vehicle’s position using its speed.
   1. Position =

self.ego\_position += self.current\_speed \* self.delta\_t

1. Calculate the distance to the lead vehicle using the lead vehicle position in the current episode and the calculate ego vehicle’s position.

distance\_to\_lead = self.current\_position[self.step\_idx] - self.ego\_position

1. Use the ego vehicle’s current speed, the reference speed, the distance to the lead vehicle, the jerk, and the current acceleration to calculate the current reward.
   1. The reward function will be explained in the next section.
2. Return the following information to the system to signify a completed step:
   1. Training Environment: Speed error, distance error, distance to the lead vehicle, jerk.
   2. Testing Environment: Speed error, distance error, distance to the lead vehicle, jerk, lead vehicle speed.

## Reward Structure

The reward function was redesigned to balance three objectives:

1. Speed Tracking: Minimize the deviation from reference speed.
2. Distance Tracking: Penalize the ego vehicle if outside of the safe following distance constraints.
3. Smooth Behavior: Penalize abrupt acceleration changes.

The reward function is made up of three components. The first component is the **speed error**. The speed error is calculated using the absolute difference between the ego vehicle’s speed and the reference speed.

speed\_error = abs(current\_speed - ref\_speed)

The second component is the **distance error**. The goal is to penalize the ego vehicle if it is too close (<5m) or too far (>30m) from the lead vehicle. However, given safety is the primary concern, there is a stronger penalty applied for the ego vehicle being too close to the lead vehicle. The distance penalty is implemented as the absolute value of the difference between the distance to the lead vehicle and the respective minimum/maximum distance being violated. However, if violating the minimum distance requirement, then the penalty is tripled.

if distance\_to\_lead < MIN\_DISTANCE:

    distance\_error = 3.0 \* abs(MIN\_DISTANCE - distance\_to\_lead)

elif distance\_to\_lead > MAX\_DISTANCE:

    distance\_error = abs(MAX\_DISTANCE - distance\_to\_lead)

else:

    distance\_error = 0

The third component is the **jerk error**. The goal is to promote smooth behavior by discouraging sudden acceleration changes for passenger comfort. This combines both the jerk and current acceleration, where the error is calculated by summing the absolute value of both values, but the current acceleration is divided by 2 to emphasize correction to the jerk value.

jerk\_error = abs(jerk) + 0.5 \* abs(accel)

Finally, all errors are combined into one final error equation, where each individual term has different weights. The distance error is the most significant safety concern, so its value was tripled in this equation. Moreover, the jerk error is the least significant safety concern, so its value was divided by 10 to emphasize the speed error and distance error corrections. Note that dividing by 10 does not fully diminish the value, as the jerk can be up to an absolute value of 4 ().

total\_error = speed\_error + 3.0 \* distance\_error + 0.1 \* jerk\_error

In implementation, the **reward** is set to be the negative value of this error value. Specifically,

reward = -error

## Performance Metrics

To evaluate the ACC system, we tracked 10 metrics during testing. They are outlined below. For these equations, represents the reward at step , represents the ego vehicle’s speed, represents the reference speed, is the distance from the ego vehicle to the lead vehicle, is the acceleration of the ego vehicle, is the change in time between steps, is the jerk, and is the total number of testing steps.

1. **Average Reward:** Represents the average reward for the ego vehicle during testing. This is useful to directly compare the training performance of each configuration. In this case, higher scores are better performing.
2. **Mean Absolute Error (MAE):** Represents the average absolute deviation between the ego vehicle’s speed and the reference speed. This is useful because it measures how closely the ego vehicle follows the reference speed. For this metric, lower values are better, and it is in units of velocity (where a minimization close to 0 m/s is preferred). The equation is shown below.
3. **Root Mean Squared Error (RMSE):** Represents the square root of the average squared speed errors. This also measures how closely the ego vehicle follows the reference speed. For this metric, lower values are better, and it is in units of velocity (m/s). Unlike MAE, larger speed errors are punished more severely. The equation is shown below.
4. **Score:** Represents the proportion of speed variance that can be explained by the model. It is useful because it quantifies how well speed trends are matched. In terms of units, it is a proportion, where a value of 1.0 is perfect. Therefore, higher values are preferred.
5. **Mean Distance:** Represents the average following distance to the lead vehicle. It is useful to evaluate average spacing behavior, where a maximization of both distance boundaries is 17.5m.
6. **Time in Safe Zone (%):** Represents the percentage of time that the ego vehicle is within the safe distance range of 5-30m. This directly measures safety compliance, making it a strong comparison metric. The higher the percentage, the better the compliance.
7. **Min/Max Distance:** Represents the closest and furthest distance to the lead vehicle during testing, respectively. This is useful to identify safety violations and emphasize potential concerns with a model’s ability to follow instructions. The worse the difference of each distance is from the safety limits, the worse the performance.
8. **Mean Jerk:** Represents the average magnitude of the jerk, or rate of change of the acceleration. This is useful to measure overall ride comfort, where a lower value is preferred.
9. **Max Jerk:** Represents the largest instantaneous jerk value observed during testing. This is useful to detect extreme discomfort events which may make a simulation unappealing to a user. For this metric, lower values are preferred.
10. **Jerk Variance:** Represents the variability in jerk values. This is useful to indicate acceleration consistency, where a lower variance is preferred.
11. **Speed Difference (Ego vs. Lead):** Represents the mean absolute speed difference between the ego and lead vehicles. This is similar to the first metric, but measures the error compared to the lead vehicle instead of the reference speed. This tests the adaptive response of the ego vehicle to the lead vehicle. For this metric, lower values are preferred, as it measures velocity (m/s).

Moreover, there will be plots generated to visualize these metrics. The plots generated included:

1. **Speed Tracking:** Ego vs. Reference vs. Lead speeds.
2. **Distance Over Time:** Visualize the distance over every time step and if it falls within the specified range.
3. **Jerk Profile:** Peaks of this graph indicate harsh acceleration and deceleration.
4. **Reward Profile:** Visualize the reward updating over time. Diverging rewards are clipped to emphasize the change of maximum reward behavior.
5. **Speed Difference:** Visualize the difference in speed between the ego and lead vehicles.

## Training Procedure

For replication purposes, the training procedure to generate every result in this experiment is outlined in this section. The provided file executes with a unique combination of parameters that affect model learning. The required terminal flags are listed below.

* **--model [int]:** Maps the integer range to the respective models .
* **--chunk\_size [int]:** Defines the episode length for training.
* **--learning\_rate [float]:** Defines the scale of parameter updating for the model.
* **--batch\_size [int]:** Controls the number of training samples used per update.
* **--buffer\_size [int]:** Defines the memory capacity for experience replay.
* **--tau [float]:** Influences how smoothly the target network updates.
* **--gamma [float]:** Sets the discount factor for future rewards.
* **--ent\_coef [float]:** Manages the balance between exploration and exploitation.
  + A value of -1.0 maps to ‘auto’ – a dynamic variable controlled by model confidence.
* **--net\_arch [str]:** Specifies the neural network size for function approximation.
  + Provided as a string of 2 comma-separated numbers (ex. “64,64”).

The batch script systematically varies these hyperparameters to analyze their effects on model performance. These results will be explored in the **Results** section

## Running the Training Script

To execute the training script, perform the following steps.

1. Open a command prompt and navigate to the project directory (containing both and ).
2. Type the following command:

# Results

This section presents the performance of the ACC system across many different ablation studies of RL algorithms and hyperparameters. Results are organized to address the assignment criteria: speed tracking, distance maintenance, and ride comfort.

Note that for each ablation study, only the BEST performing hyperparameter/algorithm has its visualizations displayed. To view a visual comparison for all configurations per ablation study,

1. Navigate to
2. Navigate to

Every visualization is combined into one PNG file for direct comparison. These diagrams will be referenced in each ablation study.

## Algorithm Ablation Study

Table 1 compares the performance of all four algorithms (SAC, PPO, TD3, DDPG) across the 11 key metrics defined in Section 4. Every other hyperparameter is frozen as its default value. The best result for each metric is bolded.

**Table 1:** RL Algorithm performance comparison.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm | SAC | PPO | TD3 | DDPG |
| Average Reward | **-2.475** | -630.989 | -9.794 | -3.670 |
| MAE (Ref. Speed) | **1.946 m/s** | 5.375 m/s | 2.787 m/s | **1.946 m/s** |
| RMSE (Ref. Speed) | **2.421 m/s** | 6.302 m/s | 3.477 m/s | 2.442 m/s |
| Score (Ref. Speed) | **0.644** | -1.416 | 0.265 | 0.637 |
| Mean Distance to Lead | 19.164 m | 14.424 m | 25.793 m | 15.354 m |
| Time in Safe Zone (%) | **95.4 %** | 4.7 % | 69.2 % | 91.0 % |
| Min Recorded Distance | -0.802 m | -278.194 m | **4.221 m** | -3.044 m |
| Max Recorded Distance | 34.587 m | 274.478 m | 57.248 m | **32.841 m** |
| Mean Jerk | **0.000 m/s^2** | **0.000 m/s^2** | 0.001 m/s^2 | -0.001 m/s^2 |
| Max Jerk | 3.733 m/s^3 | **0.051 m/s^3** | 4.000 m/s^3 | 3.962 m/s^3 |
| Jerk Variance | 1.562 m^2/s^6 | **0.000 m^2/s^6** | 4.472 m^2/s^6 | 2.200 m^2/s^6 |
| Mean Absolute Difference to Lead | 1.910 m/s | 5.751 m/s | 2.737 m/s | **1.887 m/s** |

In this experiment, I compared the four RL algorithms: SAC, PPO, TD3, and DDPG. All experiments were executed using the default hyperparameters to isolate the effect of the RL algorithm itself on ACC performance. Based on Table 1, **SAC** achieved the most favorable balance across the three evaluation categories: speed tracking, distance maintenance, and ride comfort. This selection is primarily justified using the average reward and time in safe zone metrics.

Looking at reward values, SAC had the highest average reward (-2.475), which means it learned a more stable and effective policy over the 100,000 training steps. PPO had a much worse score (-630.989), and TD3 also struggled with -9.794. DDPG was close to SAC at -3.670, but still not better. The reward graph visualization shows this clearly—SAC steadily improves while PPO barely makes progress, and TD3 has a lot of dips. DDPG also improves, but not as smoothly as SAC.

In terms of safety, SAC kept the ego vehicle within the safe following distance 95.4% of the time. That was the best result, followed by DDPG at 91.0%. PPO did the worst at just 4.7%, and TD3 reached 69.2%. The following distance plot shows that SAC stays within the safety range for almost the whole episode. PPO, on the other hand, has wild swings in distance, even going below 0, which isn’t realistic.

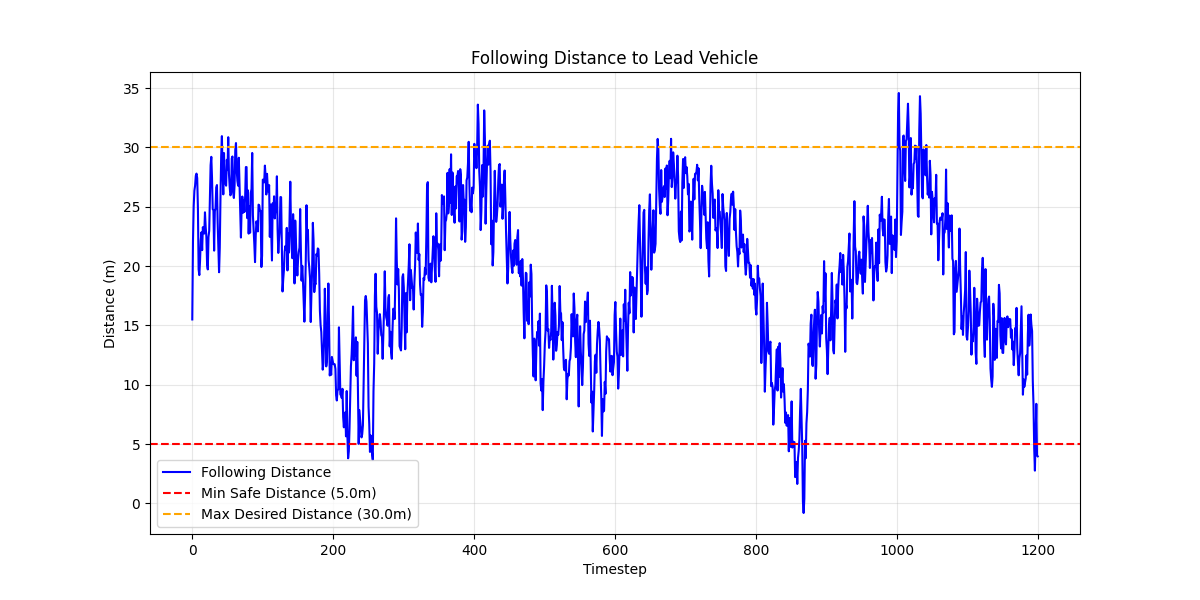
For speed tracking, SAC and DDPG both had very low mean absolute differences to the lead vehicle (1.910 m/s and 1.887 m/s). But SAC is still the better choice because it handled this more smoothly. The speed difference plots show that SAC keeps the error stable, while PPO bounces up and down a lot and TD3 has some huge spikes.

When looking at ride comfort, SAC and PPO both had a mean jerk of 0.000 , but SAC had a jerk variance of 1.562 , which shows that it had more natural acceleration changes. PPO had almost no jerk variance, meaning it barely moved or responded, which isn’t ideal. TD3 and DDPG had much higher jerk variance and max jerk values over 3.9 , which could feel uncomfortable.

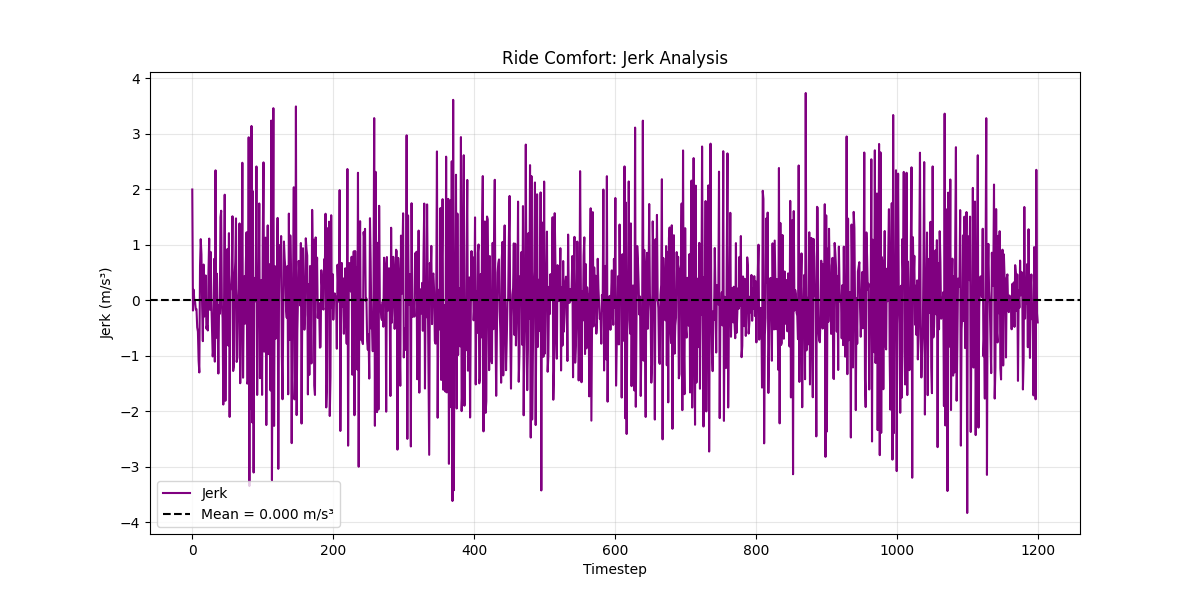
The speed performance graph shows that SAC’s ego vehicle speed closely follows the reference speed, without any sudden jumps. PPO doesn’t do well here either. Ego speed overshoots the reference by a lot, and it doesn’t match the changes in the lead vehicle speed very well.

All things considered, SAC gave the best results across reward, safety, speed tracking, and comfort. Because of that, I chose SAC as the best algorithm for this assignment. The visualizations below are all plots for the SAC algorithm.

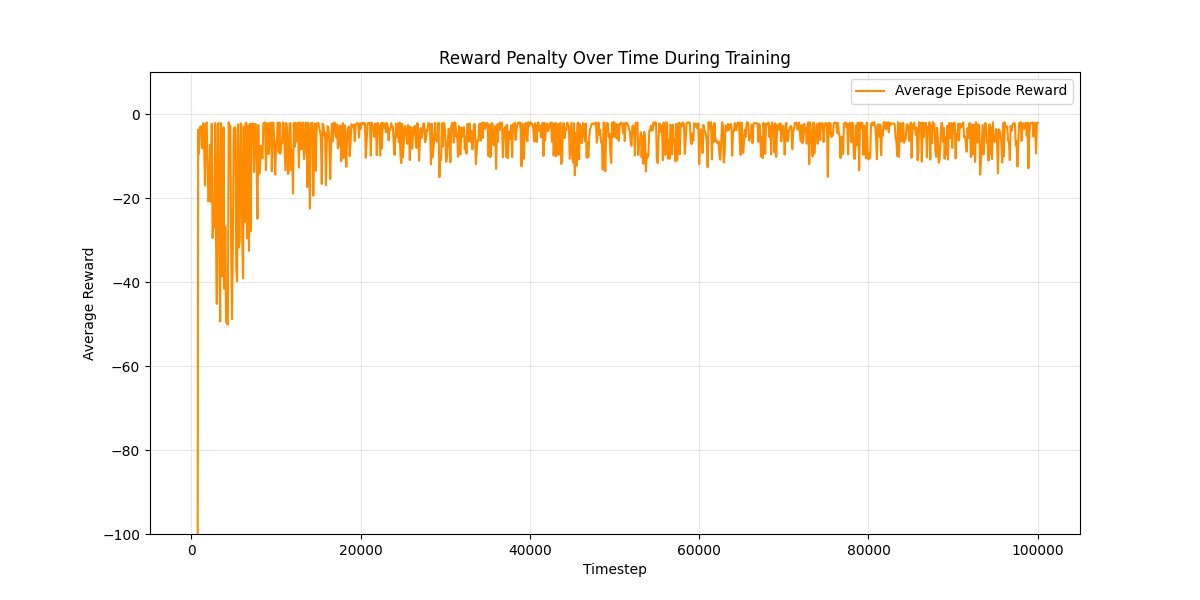
**Figure 1:** Following distance visualization across all time steps for the SAC algorithm.



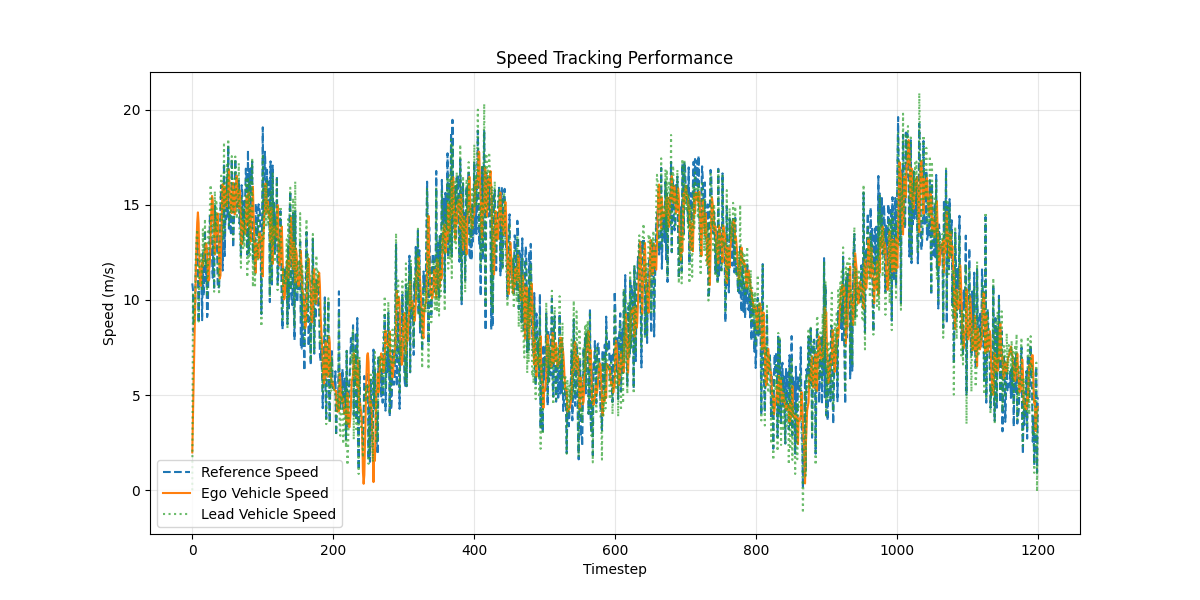
**Figure 2:** Jerk profile visualization across all time steps for the SAC algorithm.



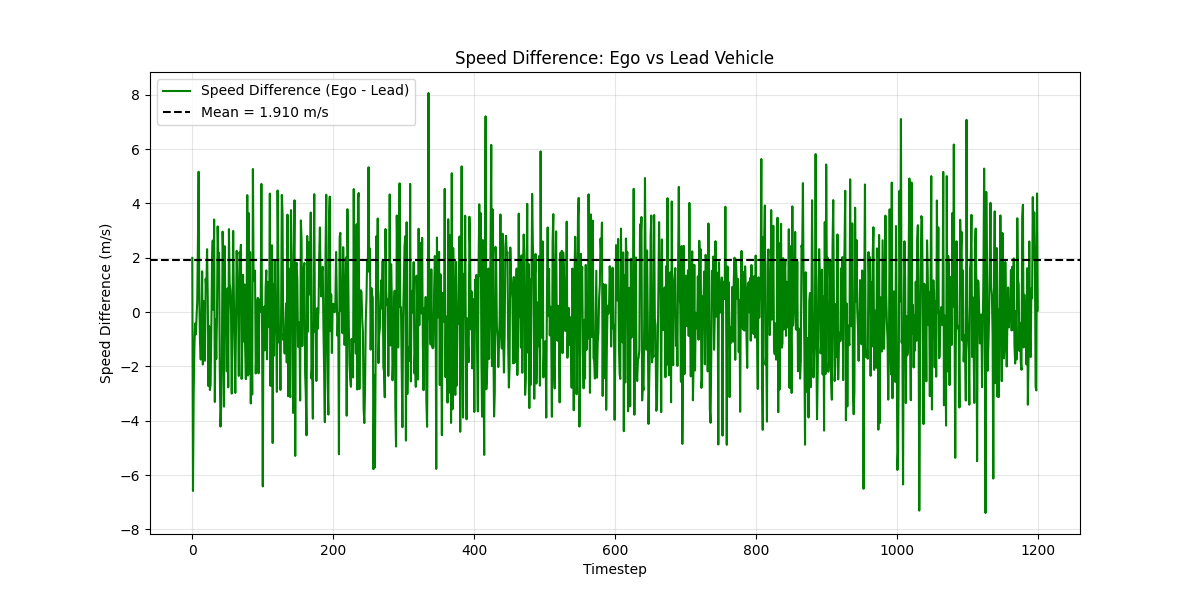
**Figure 3:** Reward penalty during training for the SAC algorithm.



**Figure 4:** Speed comparison for the ego, lead, and reference vehicles across all time steps for the SAC algorithm.



**Figure 5:** Speed difference between the ego and lead vehicles across all time steps for the SAC algorithm.



## Learning Rate Ablation Study

Table 2 compares the performance of various learning rates across the 11 key metrics defined in Section 4. Every other hyperparameter is frozen as its default value, and the SAC algorithm is used. The best result for each metric is bolded.

**Table 2:** Learning rate performance comparison.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Learning Rate | 1e-05 | 0.0001 | 0.0003 | 0.0007 | 0.001 | 0.003 |
| Average Reward | -5.604 | -2.560 | -2.475 | -7.797 | -6.557 | **-2.203** |
| MAE (Ref. Speed) | 2.112 m/s | 1.949 m/s | 1.946 m/s | **1.839 m/s** | 1.868 m/s | 1.975 m/s |
| RMSE (Ref. Speed) | 2.640 m/s | 2.422 m/s | 2.421 m/s | **2.276 m/s** | 2.342 m/s | 2.456 m/s |
| Score (Ref. Speed) | 0.576 | 0.643 | 0.644 | **0.685** | 0.666 | 0.633 |
| Mean Distance to Lead | 9.938 m | 18.730 m | 19.164 m | 21.622 m | 20.499 m | 18.314 m |
| Time in Safe Zone (%) | 82.7 % | 95.3 % | 95.4 % | 67.8 % | 74.6 % | **98.7 %** |
| Min Recorded Distance | -5.209 m | -2.298 m | -0.802 m | -0.320 m | -7.726 m | **3.370 m** |
| Max Recorded Distance | **28.308 m** | 34.061 m | 34.587 m | 46.854 m | 39.384 m | 33.299 m |
| Mean Jerk | -0.001 m/s^2 | -0.001 m/s^2 | **0.000 m/s^2** | -0.001 m/s^2 | -0.001 m/s^2 | -0.001 m/s^2 |
| Max Jerk | 3.920 m/s^3 | 3.644 m/s^3 | 3.733 m/s^3 | **3.118 m/s^3** | 3.431 m/s^3 | 3.795 m/s^3 |
| Jerk Variance | 6.007 m^2/s^6 | 2.072 m^2/s^6 | 1.562 m^2/s^6 | **1.026 m^2/s^6** | 1.836 m^2/s^6 | 2.330 m^2/s^6 |
| Mean Absolute Difference to Lead | 2.090 m/s | 1.885 m/s | 1.910 m/s | **1.782 m/s** | 1.836 m/s | 1.937 m/s |

This section compares how different learning rates affect the performance of the SAC model. Each configuration was tested with all other parameters held constant. While learning rate 0.0007 gave the best tracking metrics, I selected **0.003** as the best option overall due to its stronger safety behavior and more realistic performance for deployment.

The learning rate 0.0007 had the best technical performance. It achieved the lowest MAE (1.839 m/s), lowest RMSE (2.276 m/s), and highest score (0.685), meaning it followed the reference speed better than any other learning rate. It also had the lowest jerk variance (1.026 ) and the smallest maximum jerk (3.118 ), which indicates a smoother ride. However, its time in the safe distance zone was only 67.8%, which is the lowest of all settings. This means that even though it was accurate, it spent nearly one-third of the episode at unsafe distances, which would not be acceptable in a real driving scenario.

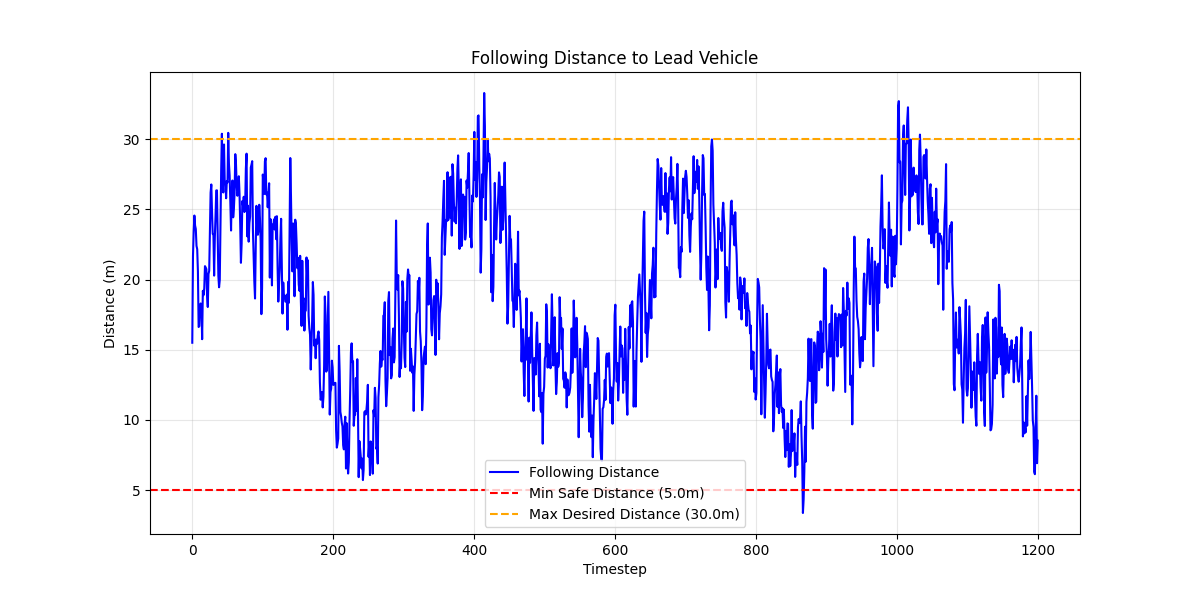
On the other hand, the learning rate 0.003 delivered the best real-world behavior. It had the highest time in the safe zone at 98.7%, and the following distance stayed mostly within the target range of 5 to 30 meters. The following distance plot shows very few violations, and both the minimum (3.370 m) and maximum (33.299 m) distances were close to acceptable. It also had the best average reward across all settings at -2.203, showing that the model learned a useful and consistent policy.

Although 0.003 had slightly worse tracking performance (MAE = 1.975 m/s, RMSE = 2.456 m/s), the error was still reasonable. The speed comparison and speed difference plots show that the ego vehicle kept up with the reference and lead vehicles without overshooting. Ride comfort was also within a usable range, with a jerk variance of 2.330 and a max jerk of 3.795 — slightly higher than 0.0007, but still smooth enough for practical use.

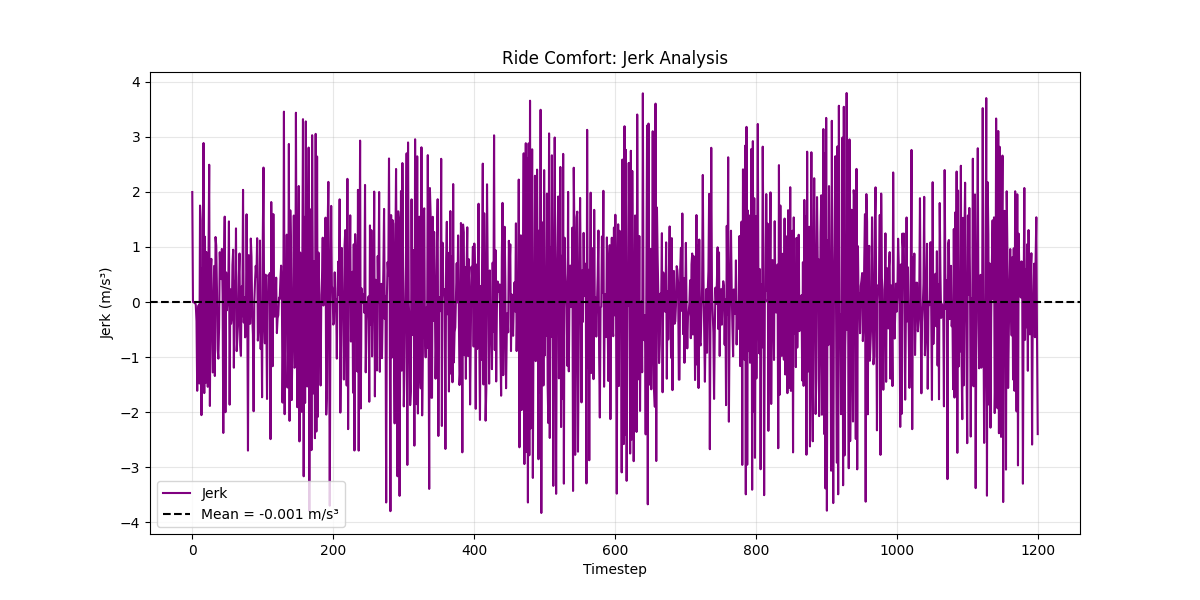
Lastly, the reward graph for 0.003 shows stable learning with high rewards and fewer drops compared to more unstable settings like 0.001 or 1e-05.

In conclusion, while 0.0007 technically performed best across many categories, its poor safety behavior makes it unsuitable for a real-world system. The learning rate 0.003 offered the best balance of safety, stability, and consistent training, making it the most realistic and reliable choice for a deployed ACC model.

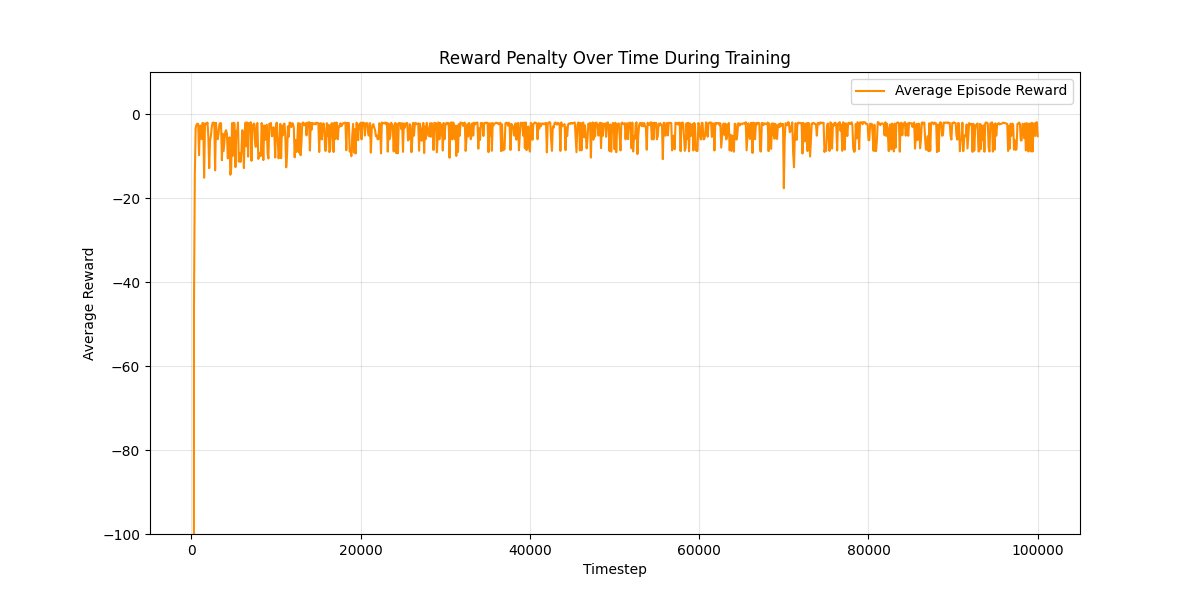
**Figure 6:** Following distance visualization across all time steps for a learning rate of 0.003.



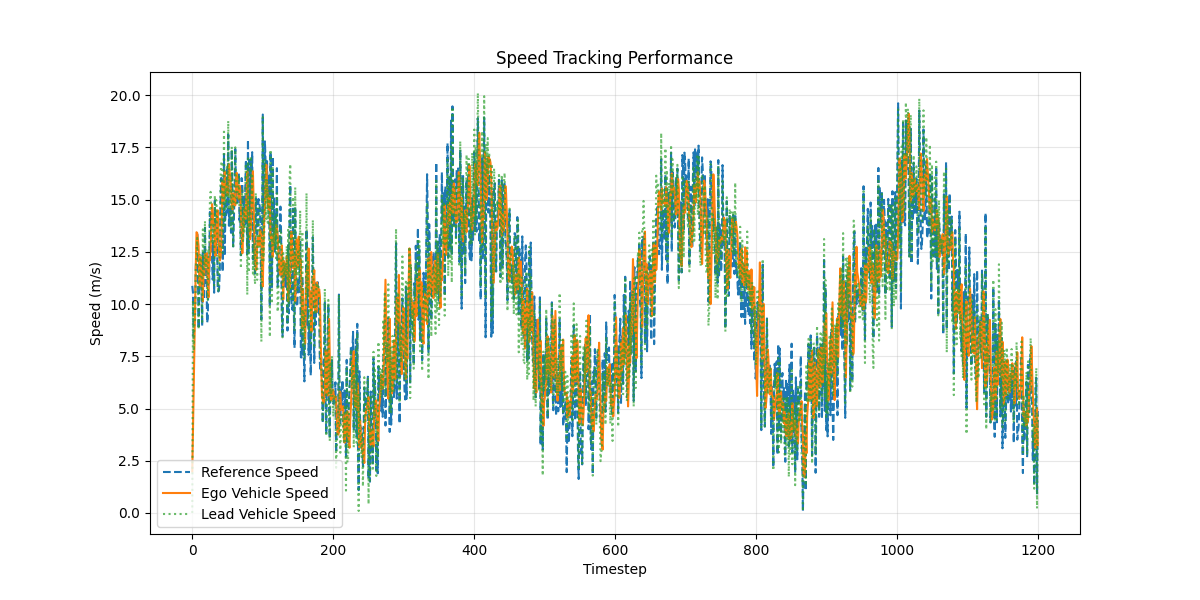
**Figure 7:** Jerk profile visualization across all time steps for a learning rate of 0.003.



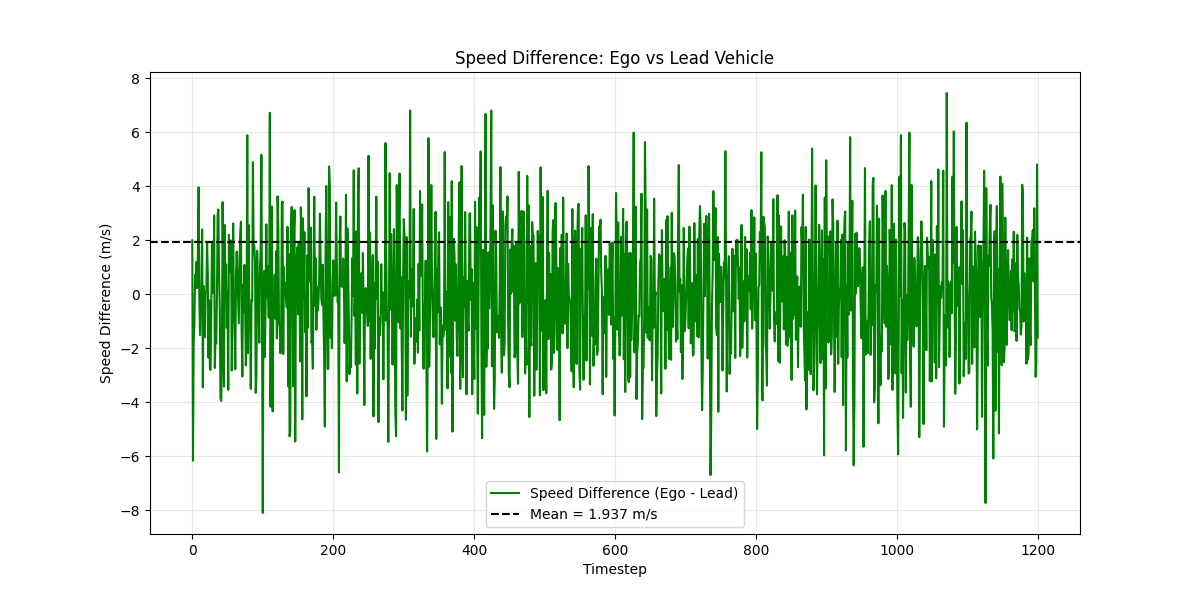
**Figure 8:** Reward penalty during training for a learning rate of 0.003.



**Figure 9:** Speed comparison for the ego, lead, and reference vehicles across all time steps for a learning rate of 0.003.



**Figure 10:** Speed difference between the ego and lead vehicles across all time steps for a learning rate of 0.003.



## Batch Size Ablation Study

Table 3 compares the performance of various batch sizes across the 11 key metrics defined in Section 4. Every other hyperparameter is frozen as its default value, and the SAC algorithm is used. The best result for each metric is bolded.

**Table 3:** Batch size performance comparison.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Batch Size | 32 | 64 | 128 | 256 | 512 |
| Average Reward | -3.499 | -2.646 | -4.497 | **-2.475** | -7.440 |
| MAE (Ref. Speed) | 1.908 m/s | 1.916 m/s | 1.906 m/s | 1.946 m/s | **1.891 m/s** |
| RMSE (Ref. Speed) | 2.390 m/s | 2.393 m/s | 2.378 m/s | 2.421 m/s | **2.353 m/s** |
| Score (Ref. Speed) | 0.653 | 0.652 | 0.656 | 0.644 | **0.663** |
| Mean Distance to Lead | 21.994 m | 16.607 m | 15.826 m | 19.164 m | 23.860 m |
| Time in Safe Zone (%) | 81.3 % | 94.6 % | 87.2 % | **95.4 %** | 63.4 % |
| Min Recorded Distance | **3.440 m** | -1.168 m | -3.505 m | -0.802 m | -2.567 m |
| Max Recorded Distance | 39.065 m | **32.324 m** | 33.967 m | 34.587 m | 44.047 m |
| Mean Jerk | -0.002 m/s^2 | -0.001 m/s^2 | -0.002 m/s^2 | **0.000 m/s^2** | -0.001 m/s^2 |
| Max Jerk | 3.677 m/s^3 | 3.817 m/s^3 | 3.868 m/s^3 | 3.733 m/s^3 | **3.320 m/s^3** |
| Jerk Variance | 2.663 m^2/s^6 | 2.342 m^2/s^6 | 2.932 m^2/s^6 | **1.562 m^2/s^6** | 2.147 m^2/s^6 |
| Mean Absolute Difference to Lead | **1.850 m/s** | 1.883 m/s | **1.850 m/s** | 1.910 m/s | 1.861 m/s |

This section compares different batch sizes to evaluate their effect on the SAC model’s training and test performance. Each batch size was tested while keeping all other settings the same. From the results in Table 3, the batch size **256** was selected as the best option overall.

In terms of reward performance, a batch size of 256 gave the highest average reward at -2.475. This shows that the model learned a more effective policy compared to the other settings. A batch size of 64 came in second at -2.646, while 512 had a much worse reward score of -7.440, which suggests instability in learning. The reward plot also shows that 256 had a smoother training curve compared to settings like 32 or 512, which had frequent drops and noise.

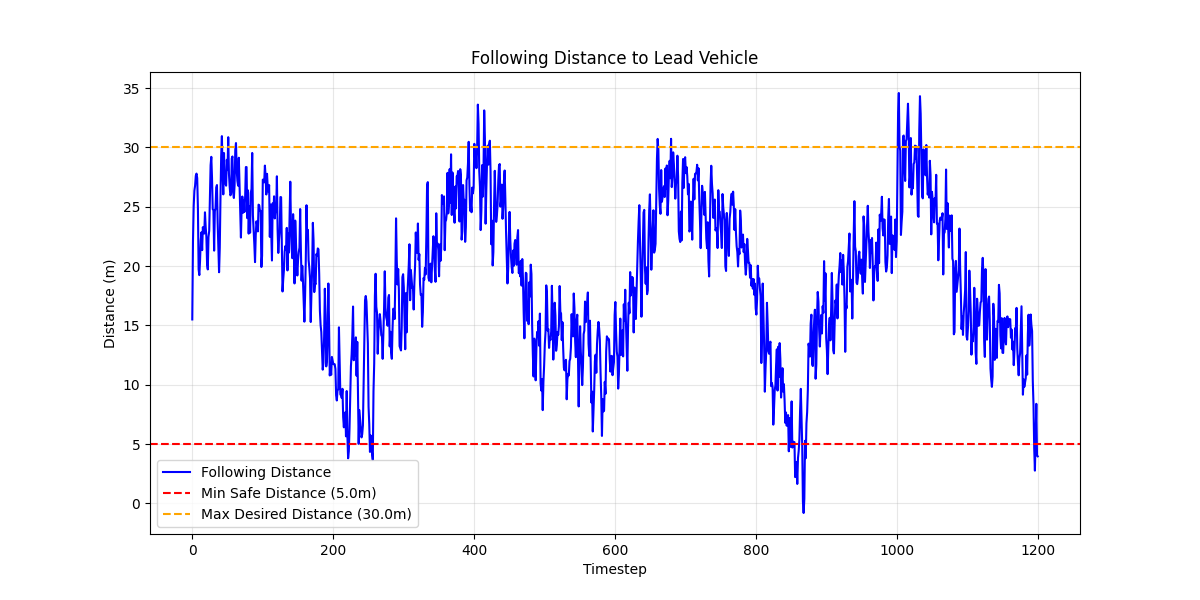
For safety, a batch size of 256 had the highest time in the safe distance zone at 95.4%. This means the ego vehicle stayed within the target 5–30 meter range nearly the entire time. The following distance chart shows that it stayed between the safety thresholds with only a few brief outliers. A batch size of 64 also performed well (94.6%), but 512 dropped significantly to just 63.4%, showing more time spent outside of safe distances.

Tracking performance across all batch sizes was fairly close. While a batch size of 512 had slightly better MAE (1.891 m/s) and RMSE (2.353 m/s), the differences were small. A batch size of 256 had MAE = 1.946 m/s and RMSE = 2.421 m/s, which is still competitive. In the speed comparison plots, a batch size of 256 shows solid overlap between the ego speed and the reference speed, without major overshoots.

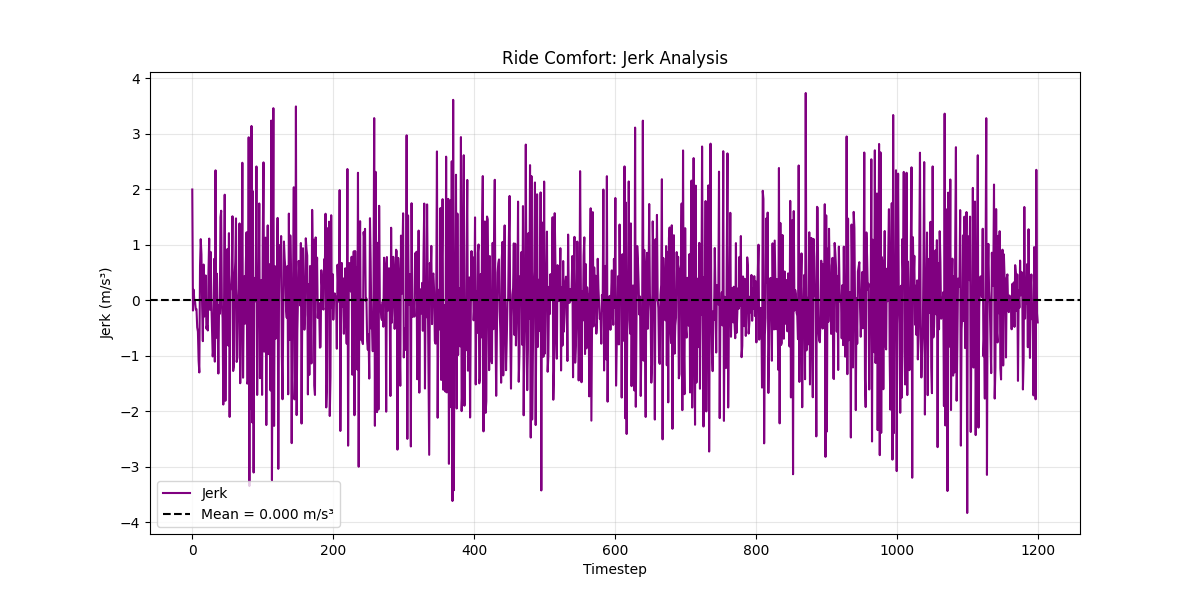
Comfort metrics (jerk) were also similar across most settings. A batch size of 256 had a mean jerk of 0.000 and jerk variance of 1.562 , which were among the best values. The batch size 512 had slightly better max jerk (3.320 ), but all jerk metrics were within a close range, and none showed major comfort issues. The jerk profile charts confirm that a batch size of 256 had relatively even and stable behavior over time.

In conclusion, while a batch size of 512 had slightly better speed tracking metrics, its poor average reward and low safe zone time make it less reliable for real use. The batch size 256 offered the best trade-off across all areas — good learning stability, high safety, and consistent comfort.

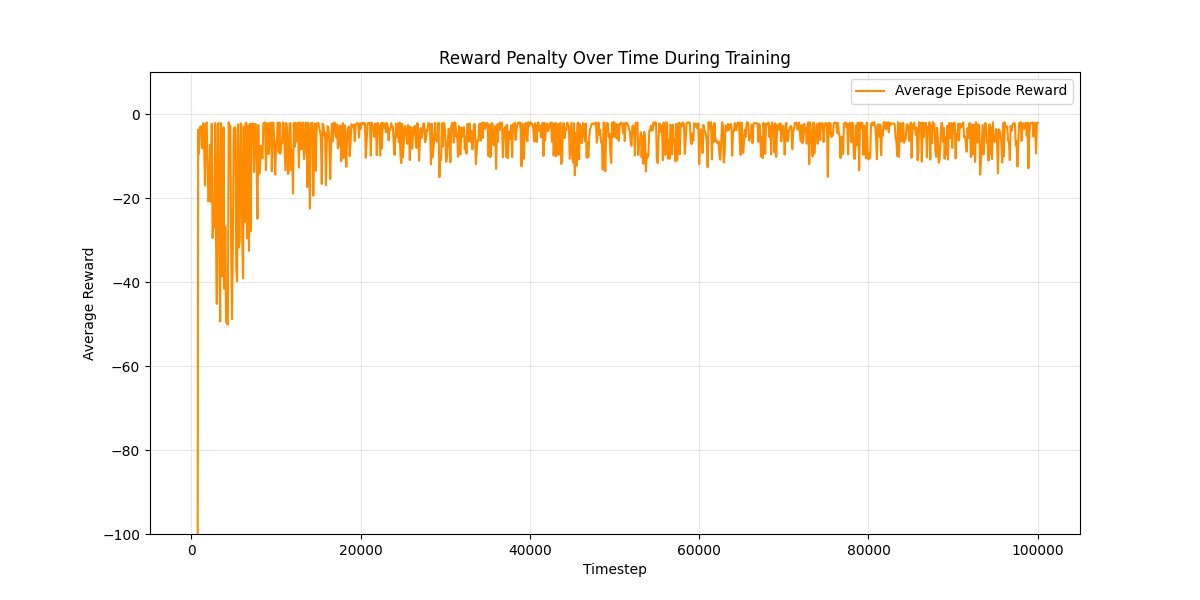
**Figure 11:** Following distance visualization across all time steps for a batch size of 256.



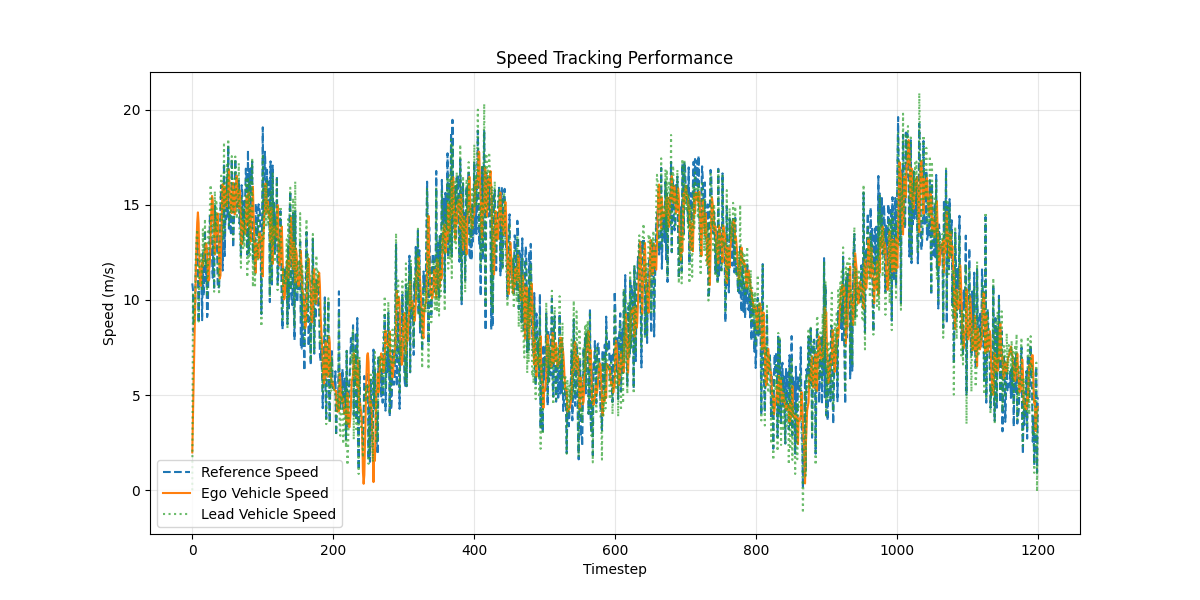
**Figure 12:** Jerk profile visualization across all time steps for a batch size of 256.



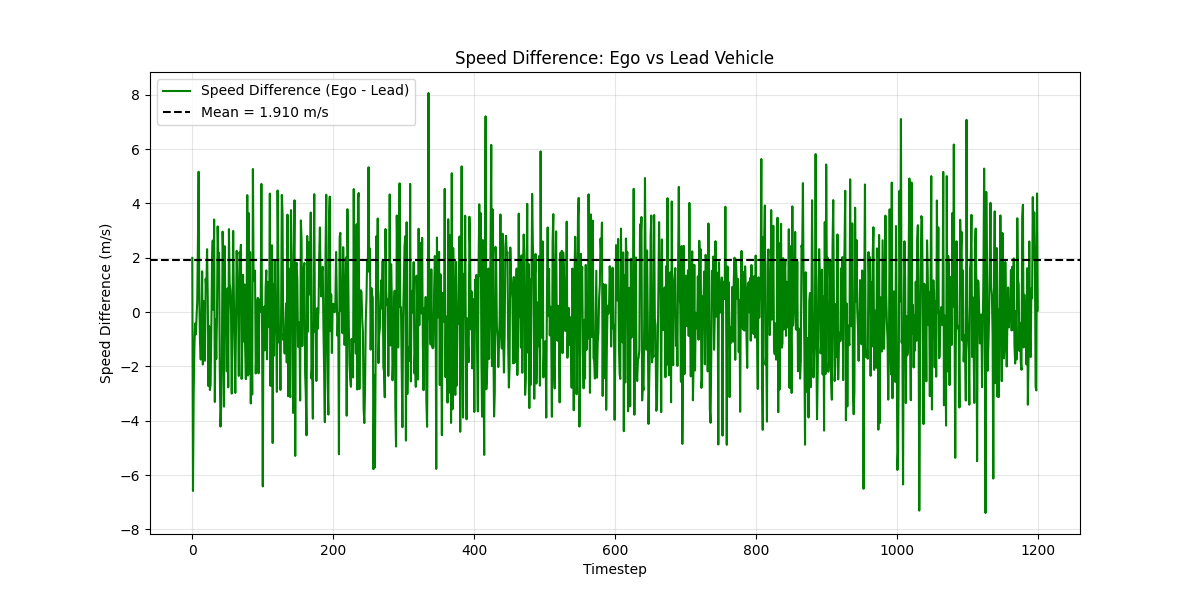
**Figure 13:** Reward penalty during training for a batch size of 256.



**Figure 14:** Speed comparison for the ego, lead, and reference vehicles across all time steps for a batch size of 256.



**Figure 15:** Speed difference between the ego and lead vehicles across all time steps for a batch size of 256.



## Buffer Size Ablation Study

Table 4 compares the performance of various buffer sizes across the 11 key metrics defined in Section 4. Every other hyperparameter is frozen as its default value, and the SAC algorithm is used. The best result for each metric is bolded.

**Table 4:** Buffer size performance comparison.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Buffer Size | 50000 | 100000 | 200000 | 500000 | 1000000 |
| Average Reward | **-2.255** | -9.498 | -2.475 | -3.114 | -10.566 |
| MAE (Ref. Speed) | 1.940 m/s | 1.855 m/s | 1.946 m/s | 2.046 m/s | **1.838 m/s** |
| RMSE (Ref. Speed) | 2.417 m/s | 2.301 m/s | 2.421 m/s | 2.550 m/s | **2.269 m/s** |
| Score (Ref. Speed) | 0.645 | 0.678 | 0.644 | 0.605 | **0.687** |
| Mean Distance to Lead | 18.358 m | 21.029 m | 19.164 m | 14.369 m | 18.729 m |
| Time in Safe Zone (%) | **97.2 %** | 61.4 % | 95.4 % | 95.7 % | 69.0 % |
| Min Recorded Distance | **1.163 m** | -2.978 m | -0.802 m | -2.181 m | -15.447 m |
| Max Recorded Distance | 34.003 m | 45.148 m | 34.587 m | **28.084 m** | 42.255 m |
| Mean Jerk | **-0.000 m/s^2** | -0.002 m/s^2 | **0.000 m/s^2** | **0.000 m/s^2** | -0.001 m/s^2 |
| Max Jerk | 3.655 m/s^3 | **3.442 m/s^3** | 3.733 m/s^3 | 3.764 m/s^3 | 3.617 m/s^3 |
| Jerk Variance | 1.211 m^2/s^6 | 1.563 m^2/s^6 | 1.562 m^2/s^6 | 1.586 m^2/s^6 | **0.973 m^2/s^6** |
| Mean Absolute Difference to Lead | 1.893 m/s | 1.810 m/s | 1.910 m/s | 2.003 m/s | **1.782 m/s** |

This section compares the effect of different buffer sizes on performance. A good buffer size should give the model access to enough training experience without slowing down learning or hurting stability. Based on the results in Table 4, a buffer size of **50,000** gave the best overall real-world performance by balancing reward, safety, and tracking accuracy.

Out of all tested settings, the buffer size 50,000 had the highest average reward (-2.255) and the highest time in the safe zone at 97.2%. These two metrics are very important for real-world driving, since they reflect how well the model learned to drive safely and effectively. While other sizes like 200,000 and 500,000 also performed well in safety (95.4% and 95.7%, respectively), they didn’t match 50,000 in reward. Other buffers like 100,000 and 1,000,000 had low reward scores (-9.498 and -10.566) and much worse safe zone time (61.4% and 69.0%), which makes them less reliable in real usage.

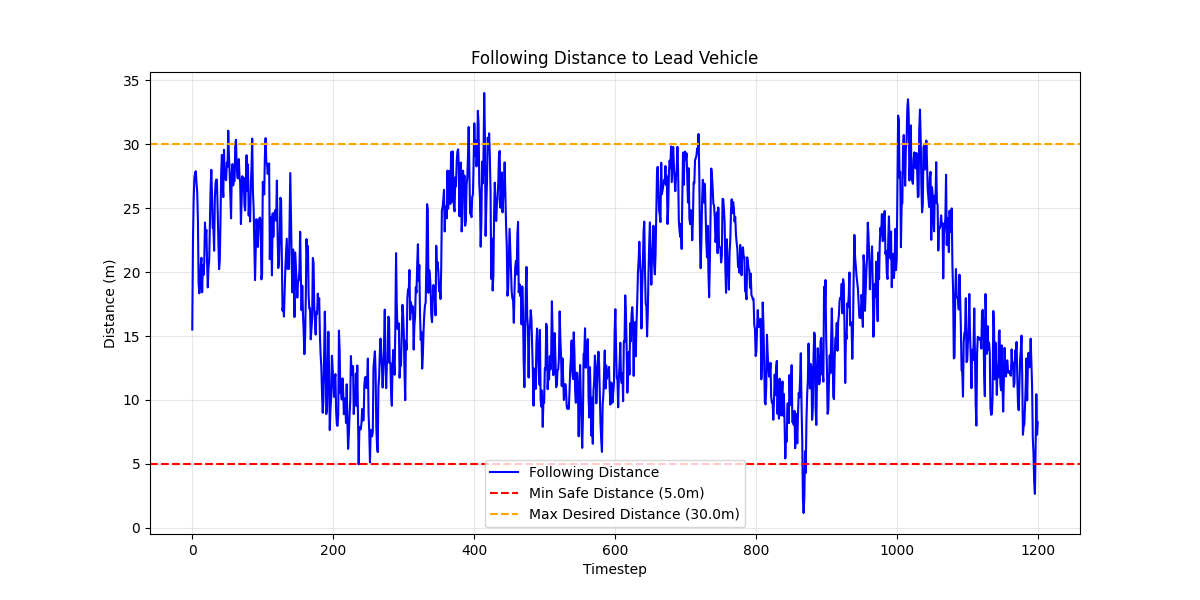
In terms of speed tracking, 50,000 had an MAE of 1.940 m/s and an RMSE of 2.417 m/s. These aren’t the absolute best values in the table, but they are still within a solid range. For comparison, buffer size 1,000,000 had slightly better speed metrics (MAE = 1.838 m/s, RMSE = 2.269 m/s), but had a terrible minimum distance (-15.447 m) and one of the worst rewards. That shows that it learned to match speed but not in a safe or useful way. The speed comparison and speed difference plots for 50,000 show that the ego vehicle follows the reference speed closely without drifting too far.

Ride comfort was also good with this setting. The jerk variance was 1.211 , which was the lowest among all configurations, and the max jerk was 3.655 — also well within acceptable limits. The jerk profile plot shows a clean and even distribution of jerk, with no sharp spikes or noisy segments.

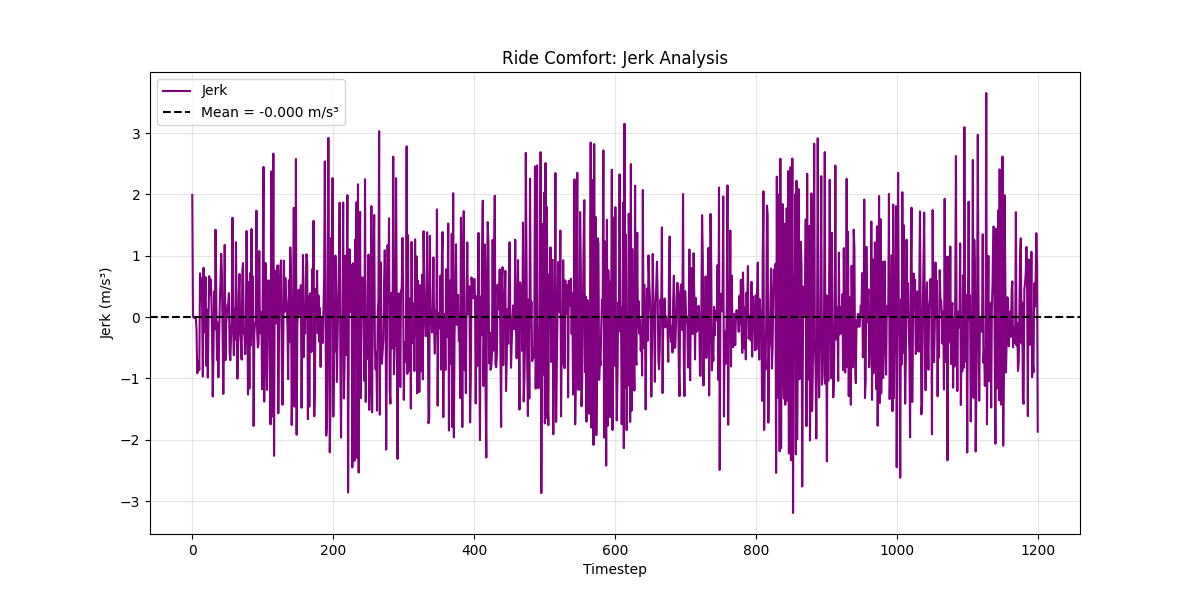
Lastly, training stability was solid. The reward plot shows that the buffer size 50,000 quickly learns a strong policy and keeps it stable without any major dips. Bigger buffers like 1,000,000 show more late-stage instability, likely because they introduce too much outdated experience into training.

In conclusion, buffer size 50,000 was the best option overall, offering the highest reward, the best safety performance, smooth ride comfort, and fast, stable training. While larger buffers had stronger tracking numbers in some cases, they weren’t reliable enough to be considered safe or usable in a real-world system.

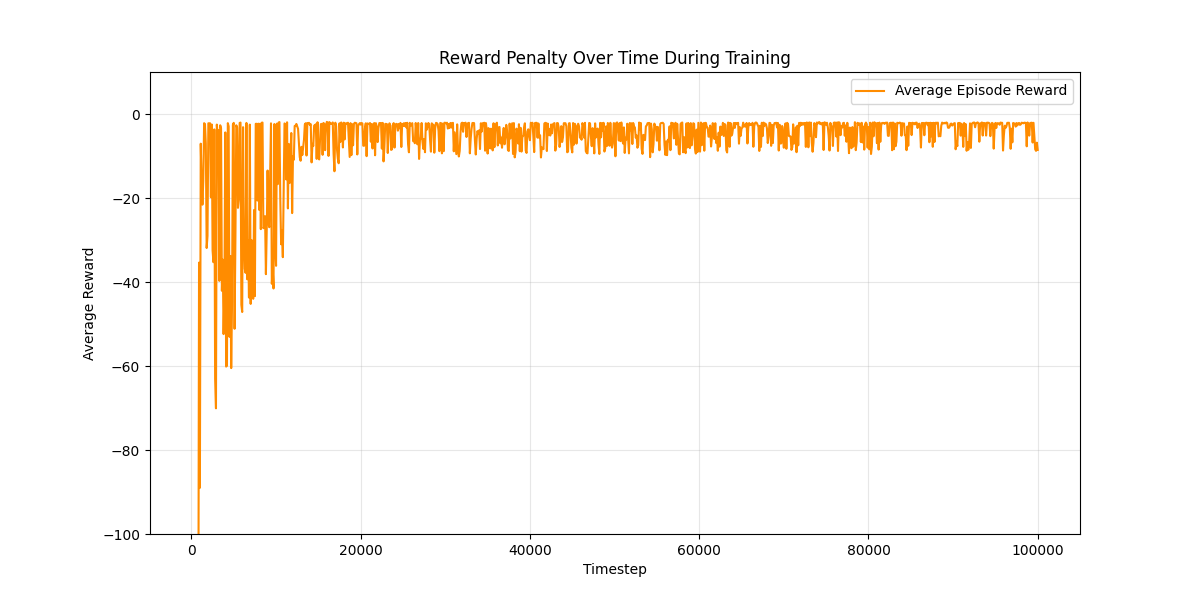
**Figure 16:** Following distance visualization across all time steps for a buffer size of 50,000.



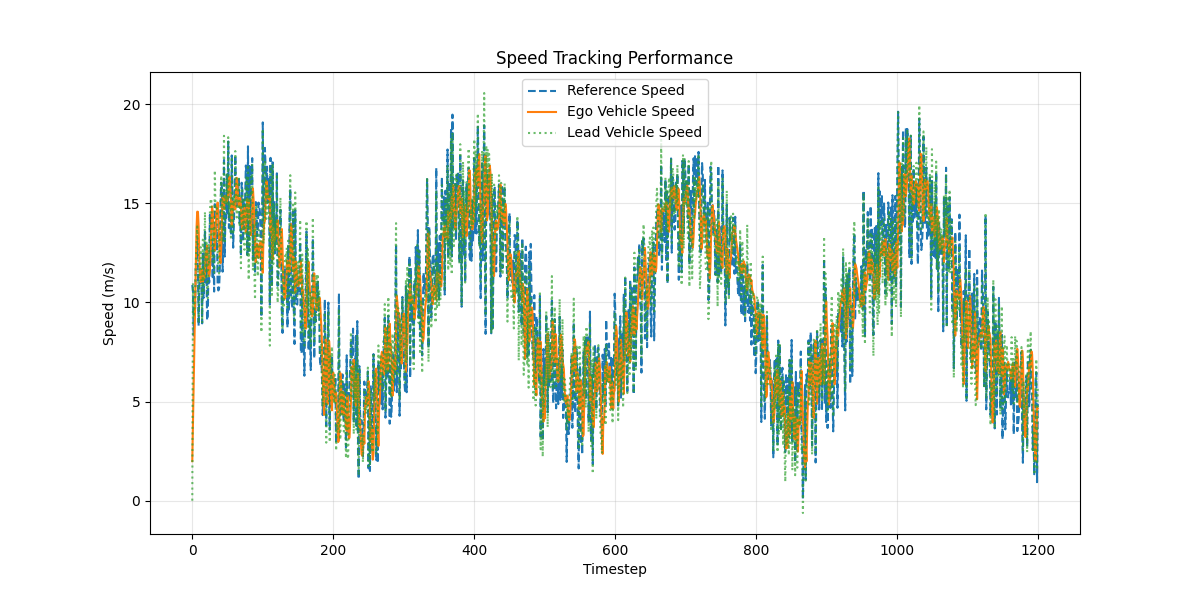
**Figure 17:** Jerk profile visualization across all time steps for a buffer size of 50,000.



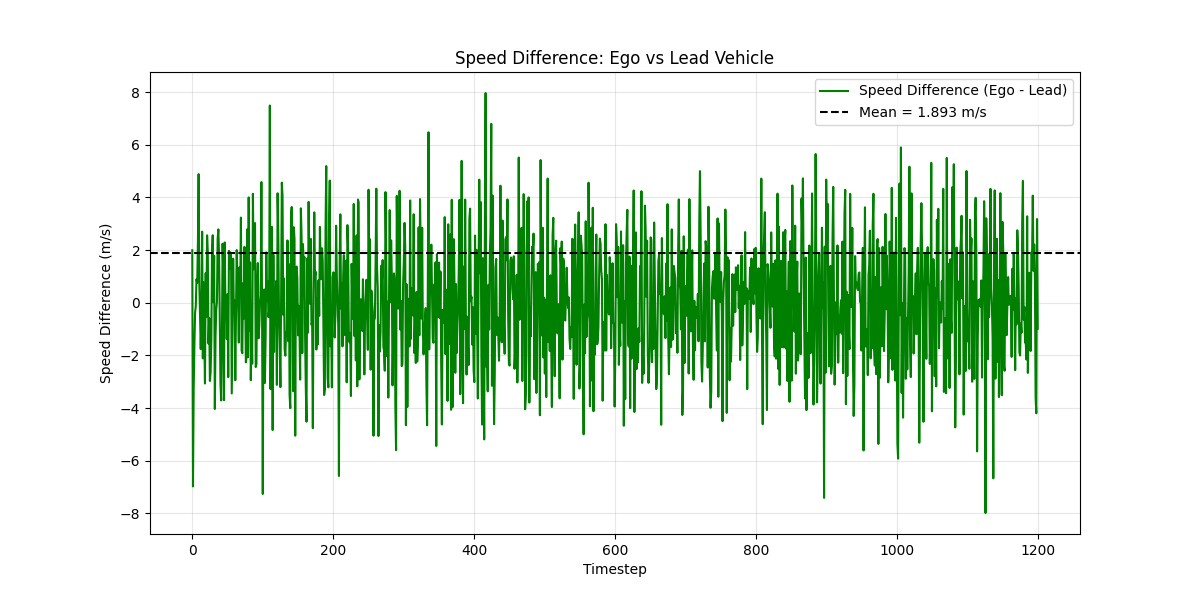
**Figure 18:** Reward penalty during training for a buffer size of 50,000.



**Figure 19:** Speed comparison for the ego, lead, and reference vehicles across all time steps for a buffer size of 50,000.



**Figure 20:** Speed difference between the ego and lead vehicles across all time steps for a buffer size of 50,000.



## Tau Ablation Study

Table 5 compares the performance of various tau values across the 11 key metrics defined in Section 4. Every other hyperparameter is frozen as its default value, and the SAC algorithm is used. The best result for each metric is bolded.

**Table 5:** Tau performance comparison.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tau | 0.0001 | 0.001 | 0.005 | 0.01 | 0.02 |
| Average Reward | -2.851 | -4.610 | **-2.475** | -7.388 | -3.282 |
| MAE (Ref. Speed) | **1.895 m/s** | 1.921 m/s | 1.946 m/s | 1.960 m/s | 2.128 m/s |
| RMSE (Ref. Speed) | **2.357 m/s** | 2.399 m/s | 2.421 m/s | 2.452 m/s | 2.644 m/s |
| Score (Ref. Speed) | **0.662** | 0.650 | 0.644 | 0.634 | 0.575 |
| Mean Distance to Lead | 19.055 m | 13.845 m | 19.164 m | 10.562 m | 12.377 m |
| Time in Safe Zone (%) | 88.6 % | 89.2 % | **95.4 %** | 78.2 % | 93.8 % |
| Min Recorded Distance | **0.302 m** | -4.362 m | -0.802 m | -6.326 m | -1.804 m |
| Max Recorded Distance | 36.666 m | 30.020 m | 34.587 m | 28.207 m | **27.026 m** |
| Mean Jerk | -0.001 m/s^2 | -0.002 m/s^2 | **0.000 m/s^2** | -0.001 m/s^2 | -0.001 m/s^2 |
| Max Jerk | **3.574 m/s^3** | 3.694 m/s^3 | 3.733 m/s^3 | 3.633 m/s^3 | 3.752 m/s^3 |
| Jerk Variance | **1.139 m^2/s^6** | 2.692 m^2/s^6 | 1.562 m^2/s^6 | 2.295 m^2/s^6 | 2.218 m^2/s^6 |
| Mean Absolute Difference to Lead | **1.826 m/s** | 1.900 m/s | 1.910 m/s | 1.905 m/s | 2.072 m/s |

Tau controls how fast the target networks in SAC update, and getting this value right is important for learning stability. I tested five values of tau and compared their effect on reward, safety, comfort, and tracking. After reviewing the data in Table 5 and the plots, I chose tau = **0.0001** as the best setting overall. While 0.005 had the highest reward (-2.475) and best time in the safe zone (95.4%), tau = 0.0001 came very close in both and gave better performance across most other categories — especially for real-world deployment.

Starting with reward, tau = 0.005 slightly outperformed with a value of -2.475, but tau = 0.0001 still did well with -2.851. The difference in training is clear in the reward plot — both learn quickly and stay stable, but 0.0001 looks like it converged stronger (smoother graph) through the episodes. Tau = 0.01 and 0.02 are clearly worse, showing more reward drops and noise.

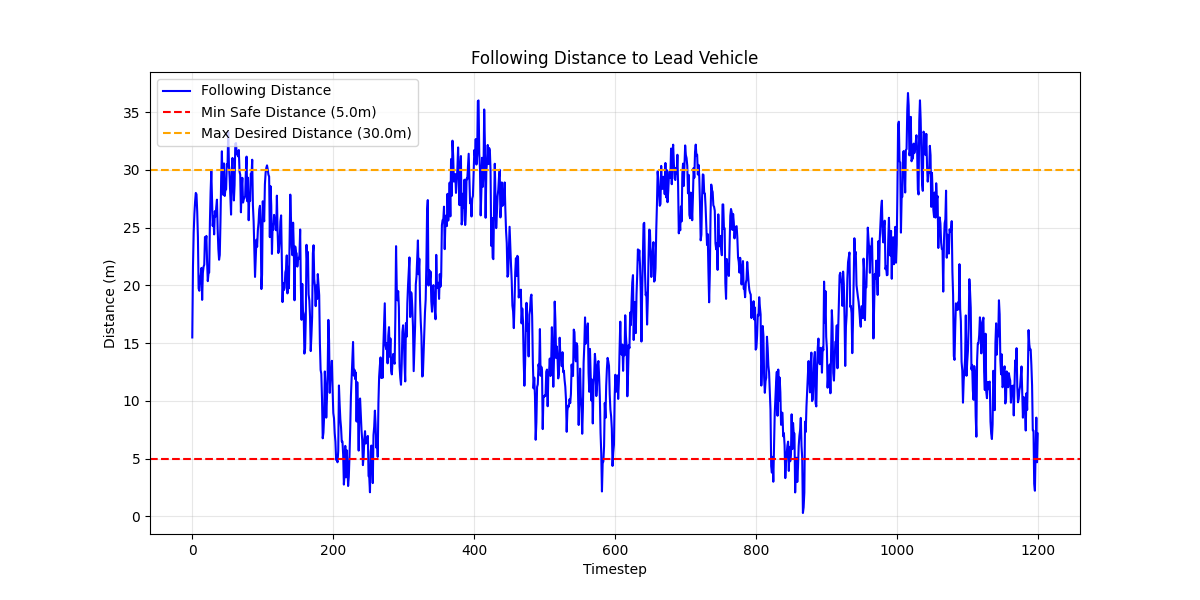
Speed tracking metrics also favor 0.0001. It had the best MAE (1.895 m/s), RMSE (2.357 m/s), and R² (0.662). However, 0.02’s score was the lowest in the group and it had a poor safety profile, so 0.0001 is the better all-around performer. The speed comparison plots show that tau = 0.0001 tracks the reference profile smoothly, without wild spikes or overshoots.

From a safety point of view, tau = 0.005 had a higher safe zone percentage (95.4%), but tau = 0.0001 still maintained a very respectable 88.6%. The following distance chart shows that tau = 0.0001 stayed within the safe distance range most of the time, and its minimum recorded distance (0.302 m) was better than all other settings; it didn’t go negative (aka surpass the lead vehicle and cause an accident), unlike tau = 0.01 (-6.326 m) or 0.001 (-4.362 m).

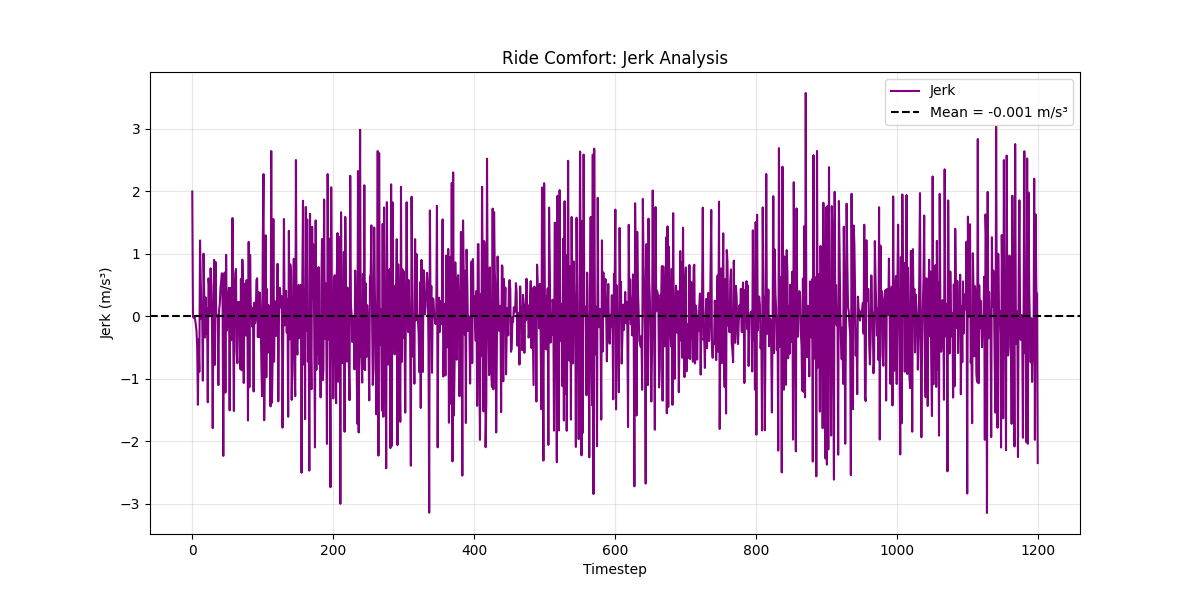
Ride comfort also favors tau = 0.0001. It had the lowest jerk variance (1.139 ), meaning the ego vehicle didn’t change acceleration too suddenly. The jerk profile plot supports this — the output is clean, with fewer sharp swings compared to tau = 0.001 or 0.01. This makes the ride feel smoother for passengers.

Putting it all together, tau = 0.0001 was the most well-rounded setting. It came close to the best in reward and safe zone time, and outperformed others in speed tracking, jerk smoothness, and minimum distance. While 0.005 was strong in safety and reward, 0.0001 gave a better overall package with more stable learning and real-world reliability.

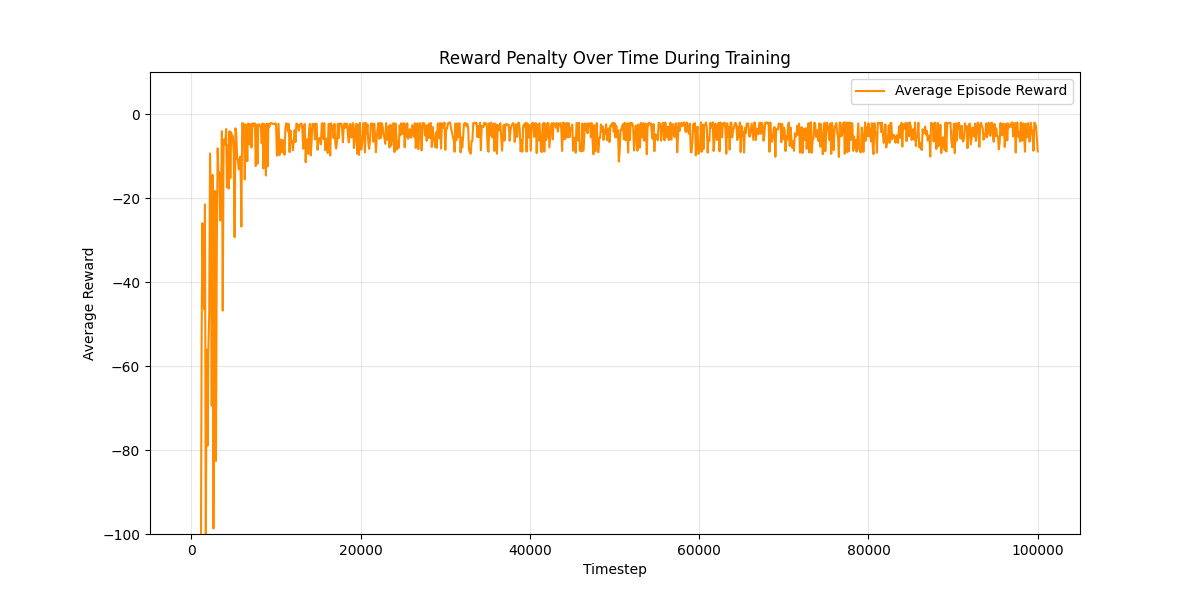
**Figure 21:** Following distance visualization across all time steps for a tau value of 0.0001.



**Figure 22:** Jerk profile visualization across all time steps for a tau value of 0.0001.



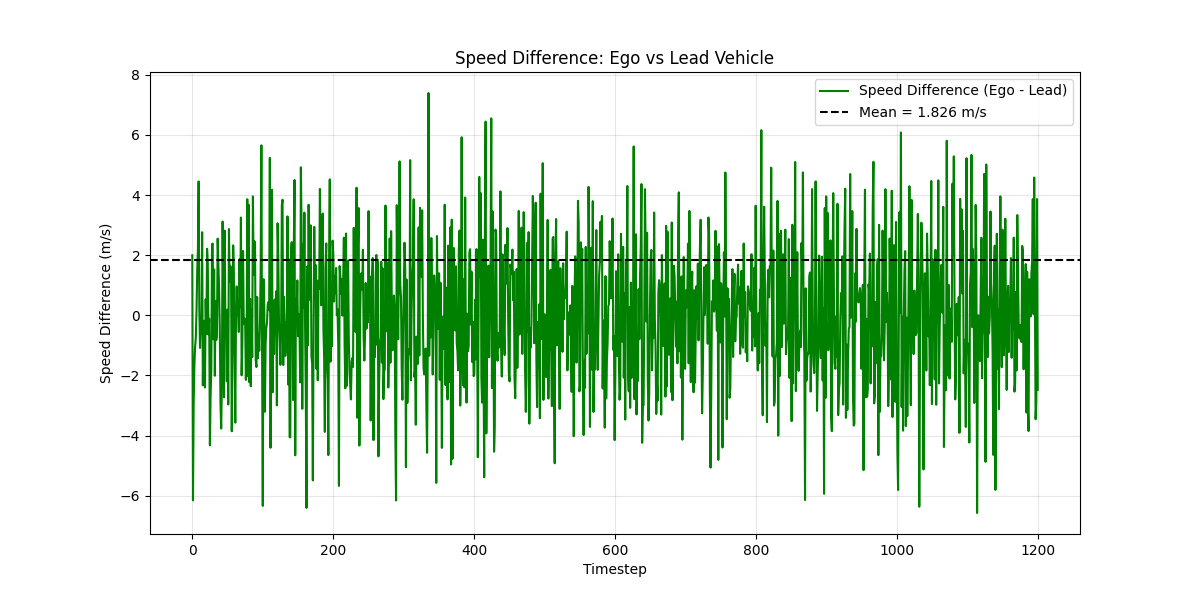
**Figure 23:** Reward penalty during training for a tau value of 0.0001.



**Figure 24:** Speed comparison for the ego, lead, and reference vehicles across all time steps for a tau value of 0.0001.



**Figure 25:** Speed difference between the ego and lead vehicles across all time steps for a tau value of 0.0001.



## Gamma Ablation Study

Table 6 compares the performance of various gamma values across the 11 key metrics defined in Section 4. Every other hyperparameter is frozen as its default value, and the SAC algorithm is used. The best result for each metric is bolded.

**Table 6:** Gamma performance comparison.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Gamma | 0.9 | 0.95 | 0.99 | 0.999 | 1.0 |
| Average Reward | -2.578 | **-2.358** | -2.475 | -7.574 | -2.474 |
| MAE (Ref. Speed) | **1.882 m/s** | 1.977 m/s | 1.946 m/s | 1.943 m/s | 1.967 m/s |
| RMSE (Ref. Speed) | **2.332 m/s** | 2.468 m/s | 2.421 m/s | 2.425 m/s | 2.450 m/s |
| Score (Ref. Speed) | **0.669** | 0.630 | 0.644 | 0.642 | 0.635 |
| Mean Distance to Lead | 18.349 m | 14.884 m | 19.164 m | 23.676 m | 19.278 m |
| Time in Safe Zone (%) | 92.0 % | **98.1 %** | 95.4 % | 67.7 % | 95.4 % |
| Min Recorded Distance | -0.465 m | **0.761 m** | -0.802 m | 0.105 m | 0.641 m |
| Max Recorded Distance | 36.069 m | **28.114 m** | 34.587 m | 46.828 m | 33.477 m |
| Mean Jerk | -0.001 m/s^2 | -0.001 m/s^2 | **0.000 m/s^2** | -0.002 m/s^2 | **0.000 m/s^2** |
| Max Jerk | **3.243 m/s^3** | 3.833 m/s^3 | 3.733 m/s^3 | 3.902 m/s^3 | 3.870 m/s^3 |
| Jerk Variance | **0.891 m^2/s^6** | 2.535 m^2/s^6 | 1.562 m^2/s^6 | 4.879 m^2/s^6 | 4.097 m^2/s^6 |
| Mean Absolute Difference to Lead | **1.829 m/s** | 1.933 m/s | 1.910 m/s | 1.945 m/s | 1.936 m/s |

For the gamma ablation study, I compared five discount factor values: 0.9, 0.95, 0.99, 0.999, and 1.0. The goal is to find the best balance between short-term and long-term decision-making. Based on the metrics and visuals, **0.95** ends up being the most practical choice, despite 0.9 performing well in a few categories.

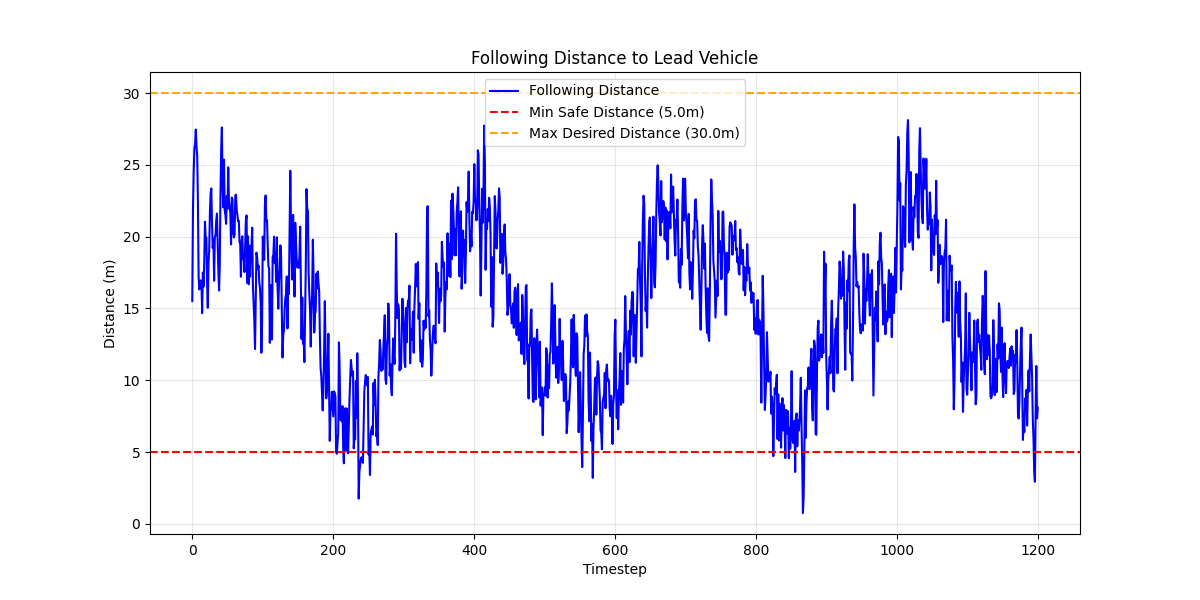
Gamma = 0.95 produced the highest percentage of time spent in the safe following distance zone (98.1%), which is the most important real-world safety metric. This value also had the highest minimum recorded distance (0.761 m), which is nonideal, but prevented collisions. While its MAE (1.977 m/s) and RMSE (2.468 m/s) are slightly worse than 0.9, this trade-off is acceptable considering its more reliable distance-keeping behavior.

Gamma = 0.9 leads in some speed metrics: lowest MAE at 1.882 m/s, lowest RMSE at 2.332 m/s, and highest score at 0.669. It also has slightly better jerk variance (0.891  vs 2.535 ), suggesting smoother ride quality. However, it only spends 92.0% of the time in the safe zone, which is 6% less than 0.95, and has a lower minimum recorded distance (−0.465 m), indicating a collision.

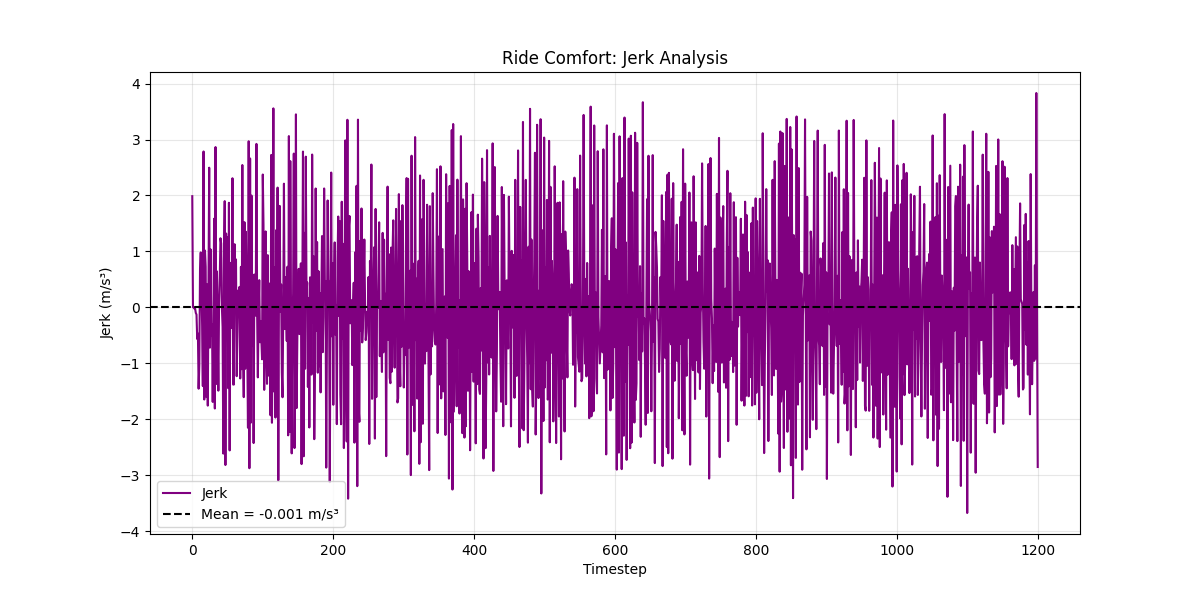
From the plots, the following distance profile for 0.95 is very stable and consistently above the safety threshold. The speed tracking performance for 0.9 is slightly cleaner, but not significantly better than 0.95. The reward curves also show that both gammas train effectively, with 0.95 having slightly more stable long-term learning.

In summary, while 0.9 offers slightly better speed tracking, 0.95 is safer and more stable overall.

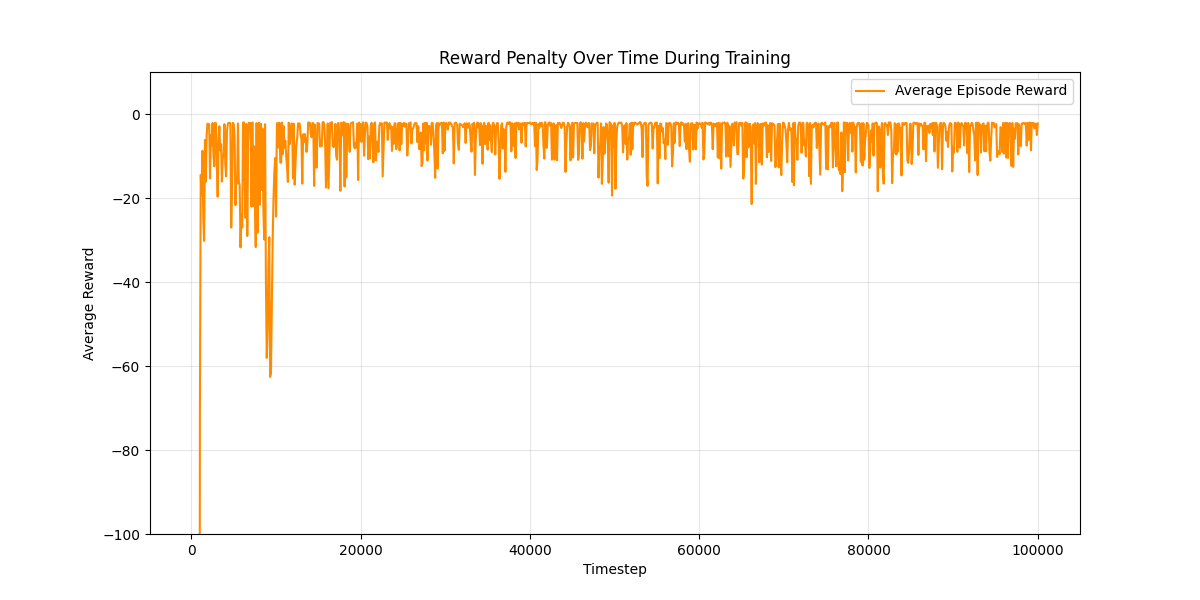
**Figure 26:** Following distance visualization across all time steps for a gamma value of 0.95.



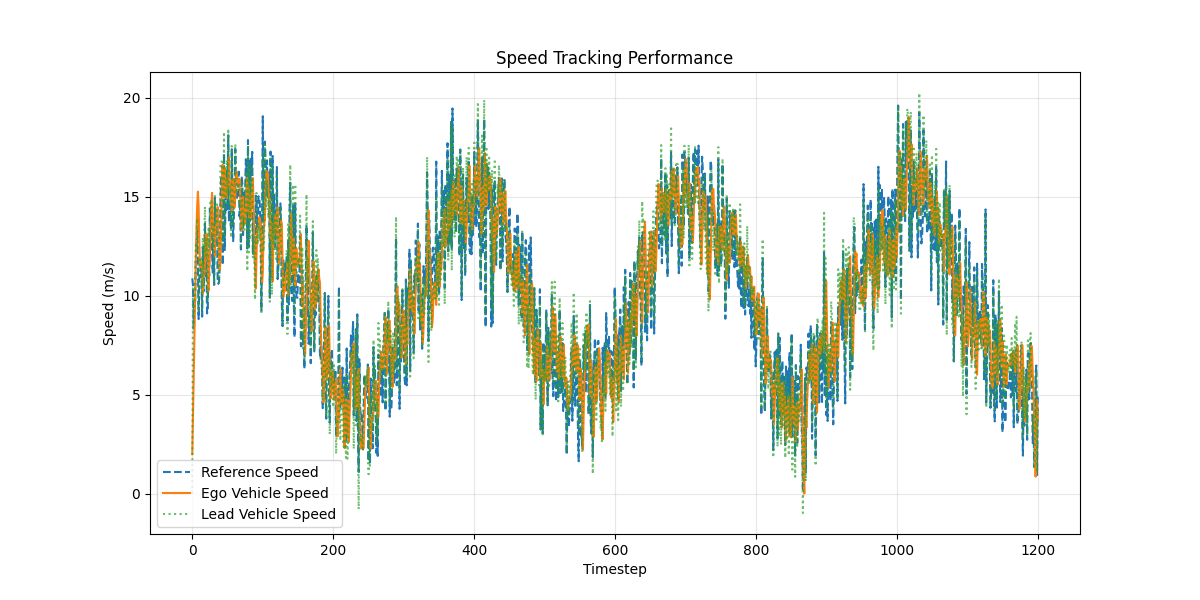
**Figure 27:** Jerk profile visualization across all time steps for a gamma value of 0.95.



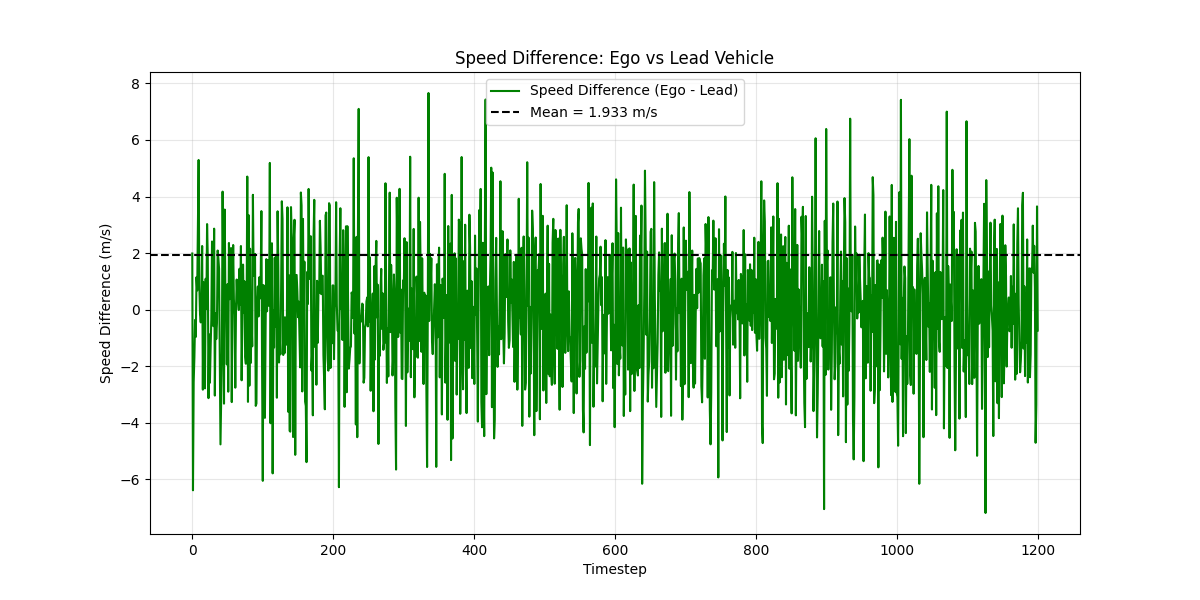
**Figure 28:** Reward penalty during training for a gamma value of 0.95.



**Figure 29:** Speed comparison for the ego, lead, and reference vehicles across all time steps for a gamma value of 0.95.



**Figure 30:** Speed difference between the ego and lead vehicles across all time steps for a gamma value of 0.95.



## Entropy Coefficient Ablation Study

Table 7 compares the performance of various entropy coefficients across the 11 key metrics defined in Section 4. Every other hyperparameter is frozen as its default value, and the SAC algorithm is used. The best result for each metric is bolded.

**Table 7:** Entropy coefficient performance comparison.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entropy Coefficient | 0.01 | 0.05 | 0.1 | auto |
| Average Reward | -2.850 | **-2.267** | -2.615 | -2.475 |
| MAE (Ref. Speed) | 1.990 m/s | 2.064 m/s | **1.906 m/s** | 1.946 m/s |
| RMSE (Ref. Speed) | 2.490 m/s | 2.562 m/s | **2.381 m/s** | 2.421 m/s |
| Score (Ref. Speed) | 0.623 | 0.601 | **0.655** | 0.644 |
| Mean Distance to Lead | 15.550 m | 16.647 m | 18.085 m | 19.164 m |
| Time in Safe Zone (%) | 95.8 % | **99.7 %** | 94.3 % | 95.4 % |
| Min Recorded Distance | -1.044 m | **1.885 m** | 0.589 m | -0.802 m |
| Max Recorded Distance | 29.651 m | **29.356 m** | 34.502 m | 34.587 m |
| Mean Jerk | -0.002 m/s^2 | -0.002 m/s^2 | -0.002 m/s^2 | **0.000 m/s^2** |
| Max Jerk | 4.000 m/s^3 | 4.000 m/s^3 | 3.804 m/s^3 | **3.733 m/s^3** |
| Jerk Variance | 4.757 m^2/s^6 | 2.200 m^2/s^6 | 2.613 m^2/s^6 | **1.562 m^2/s^6** |
| Mean Absolute Difference to Lead | 1.956 m/s | 2.007 m/s | **1.894 m/s** | 1.910 m/s |

In this ablation study, the entropy coefficient was varied to assess how exploration influences learning stability and overall ACC performance. The tested values included 0.01, 0.05, 0.1, and 'auto', with all other hyperparameters held constant. While all configurations yielded relatively competitive results, the entropy coefficient of **0.05** demonstrated the strongest overall performance across both safety and comfort metrics.

The most important metric for this section was the time in the safe zone, which directly measures how well the agent maintains a safe following distance. The model with entropy = 0.05 achieved the highest value of 99.7%, significantly higher than the others. In the corresponding distance plot, this configuration consistently stayed within the safe distance boundary throughout the test. In contrast, both 0.1 and ‘auto’ occasionally dipped toward the unsafe zone, especially in early or mid-range steps.

Comfort also played a major role in selection. The jerk profile for entropy = 0.05 exhibited moderate variability, and although its jerk variance (2.200 ) was not the lowest, it was still much better than 0.01 (4.757 ). This compromise gave a smoother ride than 0.01 while still exploring effectively during training. The 'auto' setting had the lowest jerk variance (1.562), but it underperformed on safety, recording a lower time in the safe zone (95.4%) and a negative minimum distance (−0.802 m).

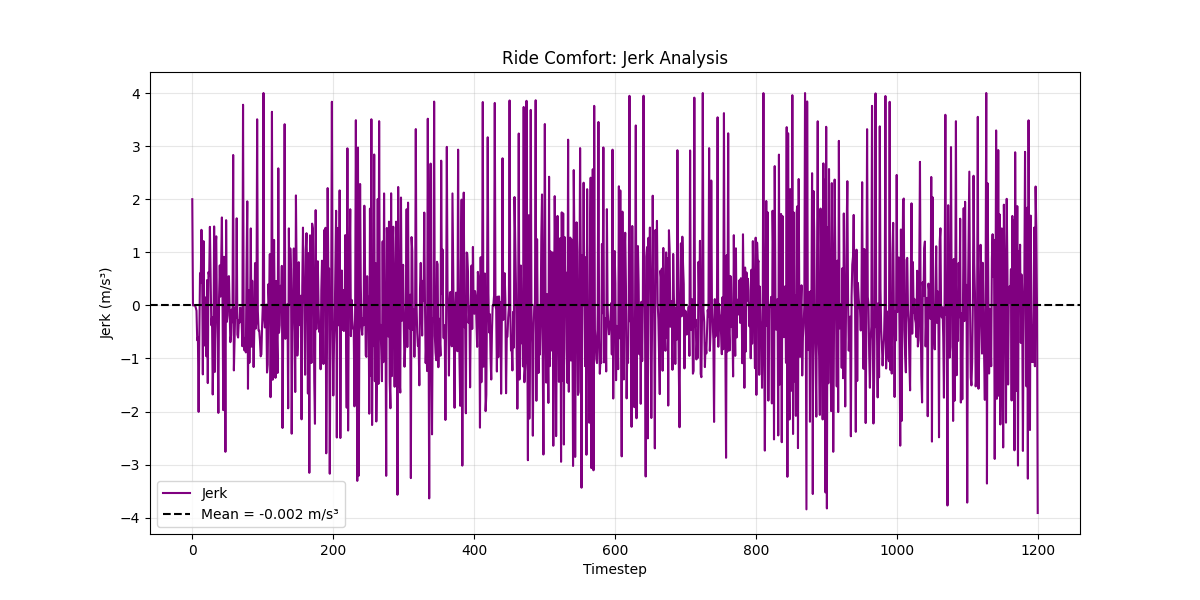
In terms of speed tracking, 0.1 achieved the lowest MAE (1.906 m/s) and RMSE (2.381 m/s), indicating closer alignment with the reference speed. However, the lead-following behavior was slightly more aggressive, with higher jerk values and a larger number of dips toward the unsafe following ranges. The difference plot confirms this, with entropy = 0.1 showing more frequent oscillations in speed relative to the lead vehicle.

Overall, entropy = 0.05 produced the most balanced performance: excellent safety, stable distance behavior, and acceptable ride comfort, even if not optimal in every metric. Visually, the results backed this up with smooth tracking curves, low jerk spikes, and reward convergence indicating steady learning.

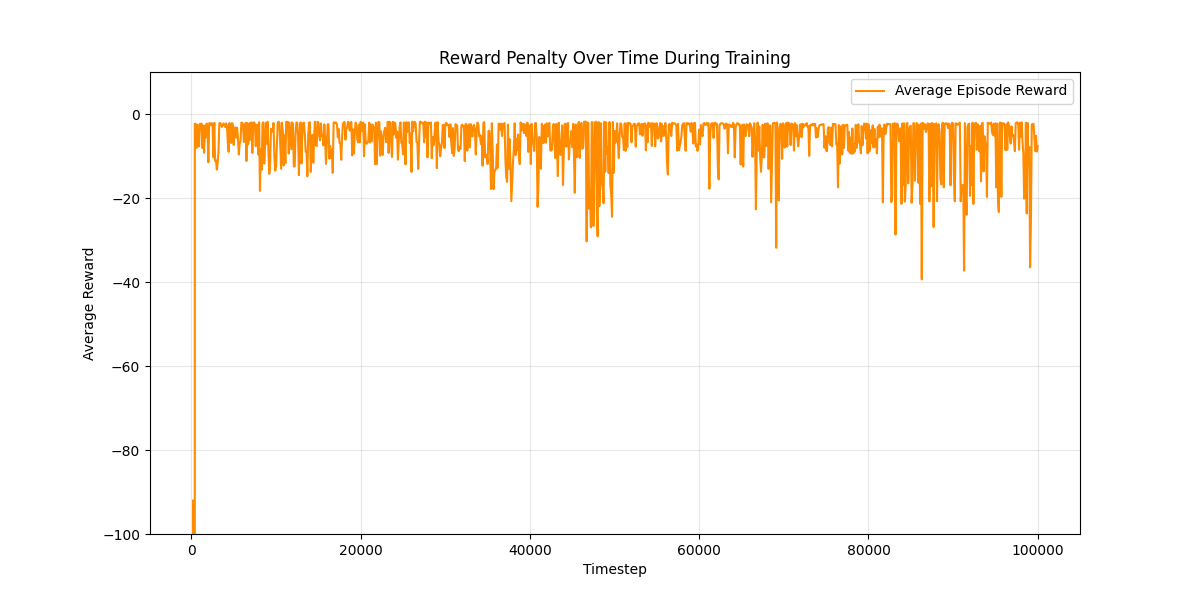
**Figure 31:** Following distance visualization across all time steps for an entropy coefficient value of 0.05.



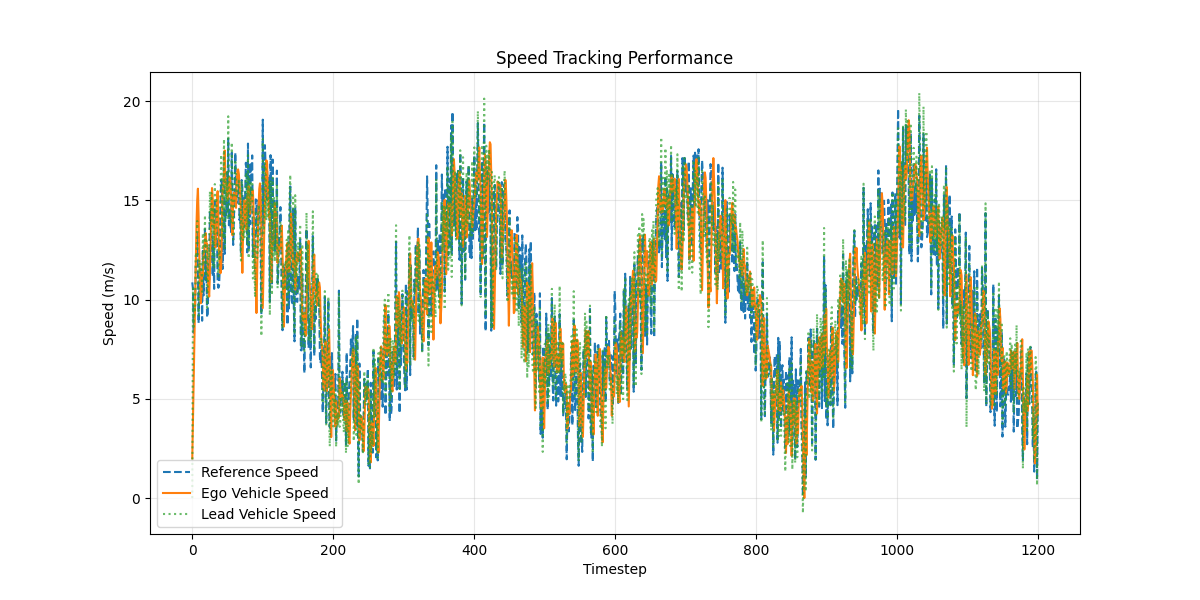
**Figure 32:** Jerk profile visualization across all time steps for an entropy coefficient value of 0.05.



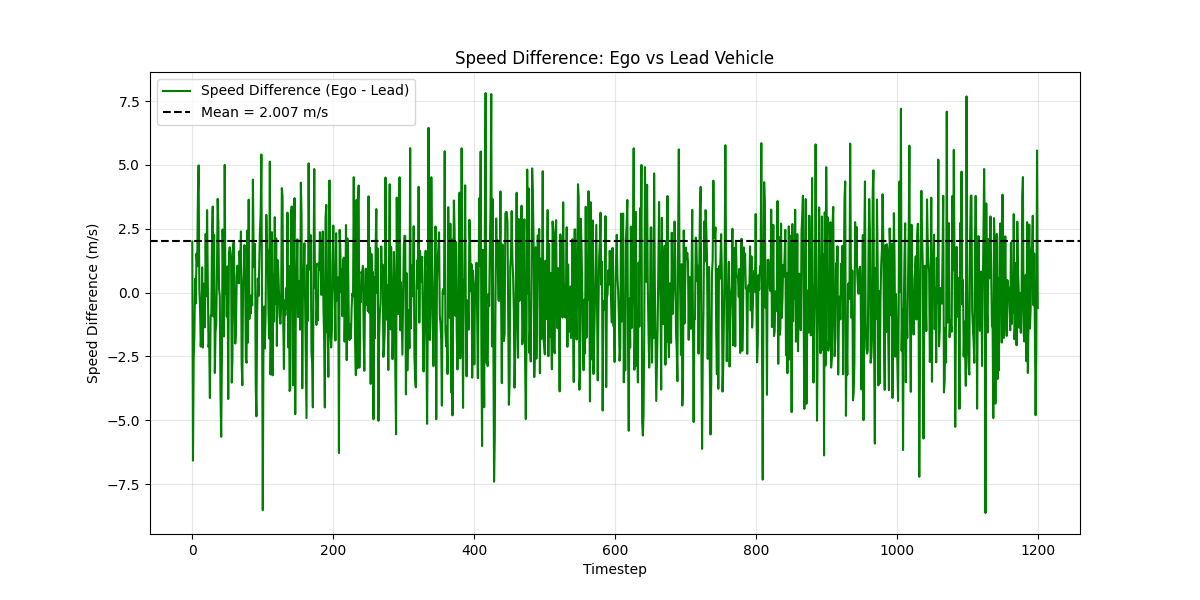
**Figure 33:** Reward penalty during training for an entropy coefficient value of 0.05.



**Figure 34:** Speed comparison for the ego, lead, and reference vehicles across all time steps for an entropy coefficient value of 0.05.



**Figure 35:** Speed difference between the ego and lead vehicles across all time steps for an entropy coefficient value of 0.05.



## Network Architecture Ablation Study

Table 8 compares the performance of various network architectures across the 11 key metrics defined in Section 4. Every other hyperparameter is frozen as its default value, and the SAC algorithm is used. The best result for each metric is bolded.

**Table 8:** Network architecture performance comparison.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Network Architecture | 64x64 | 128x128 | 256x256 | 512x512 |
| Average Reward | -3.083 | **-2.265** | -2.475 | -2.355 |
| MAE (Ref. Speed) | 1.997 m/s | 2.042 m/s | **1.946 m/s** | 2.016 m/s |
| RMSE (Ref. Speed) | 2.486 m/s | 2.547 m/s | **2.421 m/s** | 2.512 m/s |
| Score (Ref. Speed) | 0.624 | 0.605 | **0.644** | 0.616 |
| Mean Distance to Lead | 14.210 m | 17.876 m | 19.164 m | 18.109 m |
| Time in Safe Zone (%) | 93.8 % | **99.2 %** | 95.4 % | 97.8 % |
| Min Recorded Distance | -1.732 m | **4.169 m** | -0.802 m | 0.705 m |
| Max Recorded Distance | **28.566 m** | 31.467 m | 34.587 m | 31.888 m |
| Mean Jerk | **0.000 m/s^2** | -0.002 m/s^2 | **0.000 m/s^2** | -0.001 m/s^2 |
| Max Jerk | **3.261 m/s^3** | 3.972 m/s^3 | 3.733 m/s^3 | 3.393 m/s^3 |
| Jerk Variance | **1.354 m^2/s^6** | 3.028 m^2/s^6 | 1.562 m^2/s^6 | 1.712 m^2/s^6 |
| Mean Absolute Difference to Lead | 1.932 m/s | 2.010 m/s | **1.910 m/s** | 1.977 m/s |

This ablation examines how neural network size affects the ACC agent’s ability to learn smooth, safe, and accurate driving behavior. Four architectures were tested: 64x64, 128x128, 256x256, and 512x512. While both 128x128 and 256x256 gave strong results, **128x128** emerged as the most balanced configuration when considering safety, comfort, and stability.

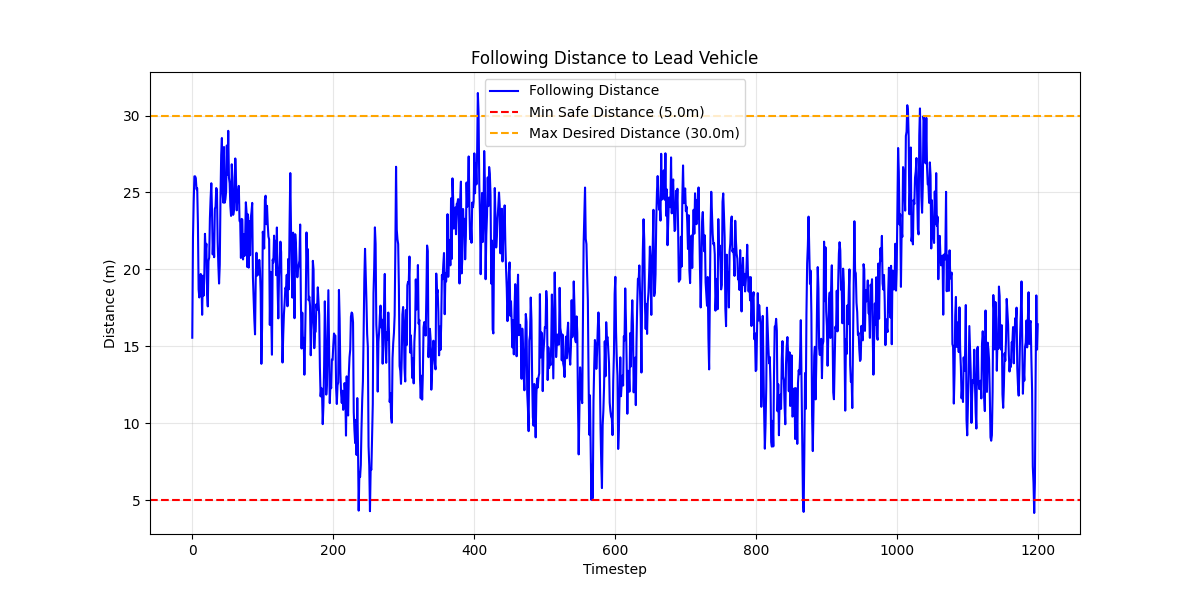
The key metric for this comparison is the time in safety zone, which directly reflects distance maintenance. The 128x128 network achieved 99.2%, clearly outperforming the others. While 256x256 was still solid at 95.4%, the 128x128 agent maintained safe distances more consistently throughout the entire sequence (seen in the corresponding distance plot), rarely dipping below 5m. Also importantly, its minimum recorded distance was 4.169m, which is significantly better than the others.

From a comfort standpoint, jerk analysis shows that 128x128 had slightly higher variance (3.028 ) compared to 256x256 (1.562), but still within acceptable range. Speed tracking (MAE of 2.042 m/s and RMSE of 2.547 m/s) wasn't the absolute best, but it was stable and showed minimal over-correction. In the speed comparison plots, both 128x128 and 256x256 closely follow the reference signal, though 256x256 had slightly tighter tracking.

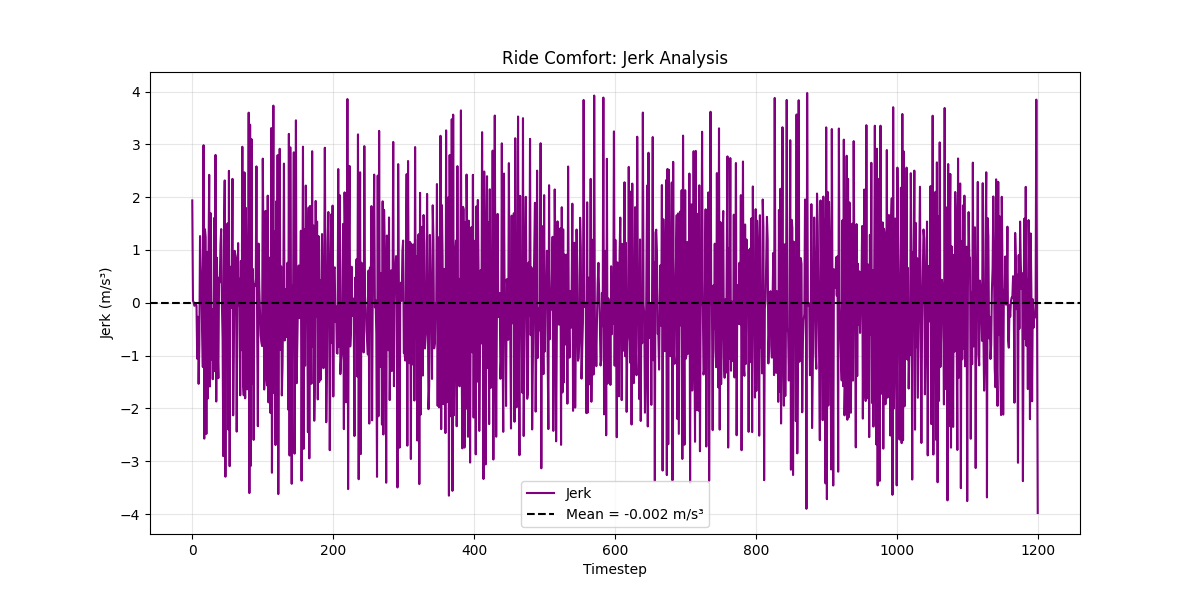
Despite 256x256 scoring better in terms of MAE, RMSE, and , its behavior was more erratic at the edges—particularly in maintaining distance, as shown by its dip below 5m and higher maximum following distance. The 128x128 model showed better generalization with smoother and more stable reward convergence during training, as visible in the reward penalty plots.

All considered, 128x128 is selected as the best architecture. It delivered the safest following profile while still offering strong speed tracking and reasonable jerk.

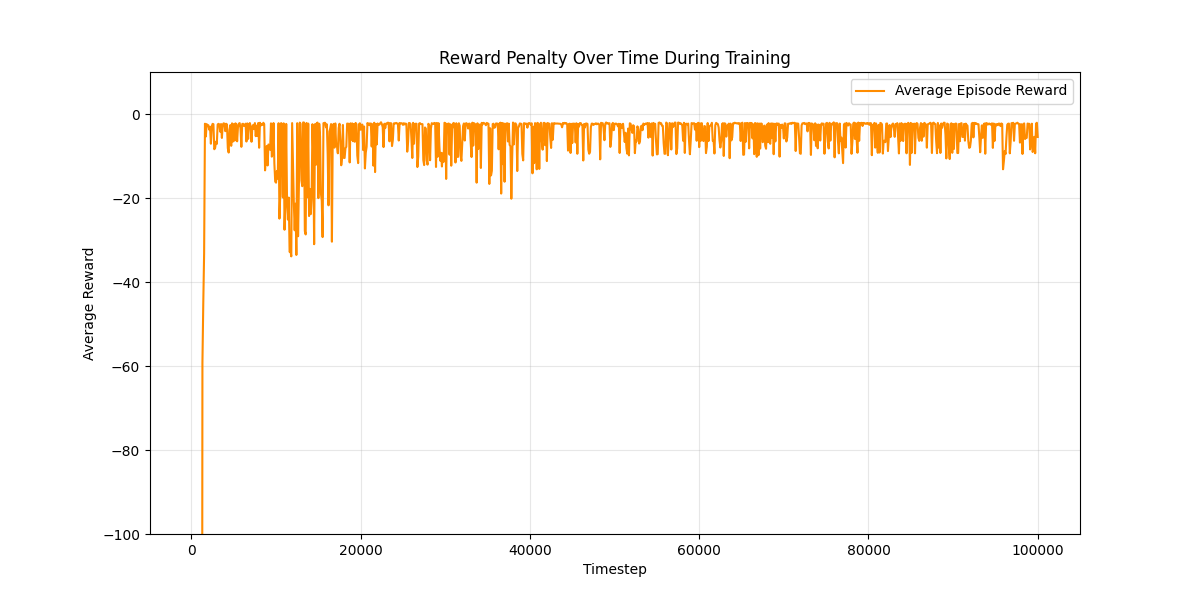
**Figure 36:** Following distance visualization across all time steps for a network architecture of 128x128.



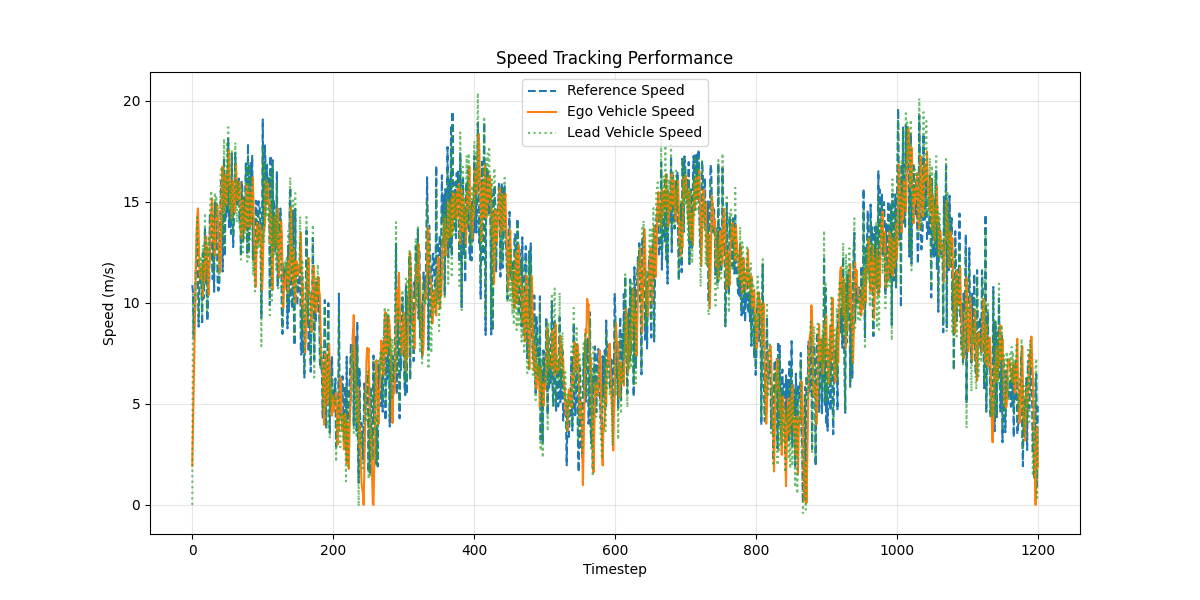
**Figure 37:** Jerk profile visualization across all time steps for a network architecture of 128x128.



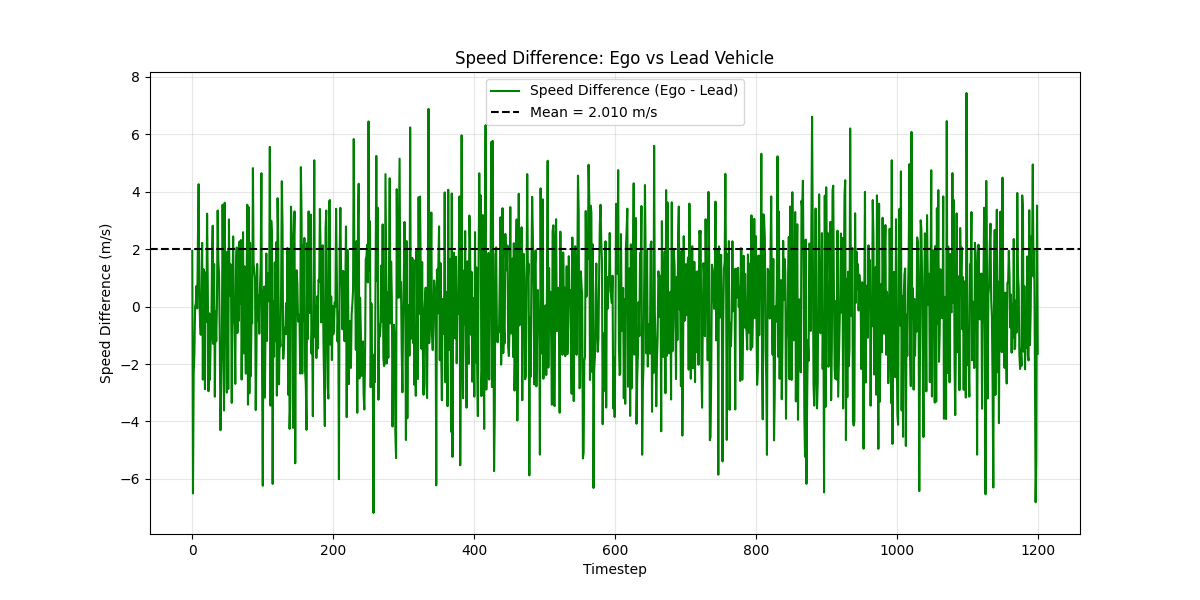
**Figure 38:** Reward penalty during training for a network architecture of 128x128.



**Figure 39:** Speed comparison for the ego, lead, and reference vehicles across all time steps for a network architecture of 128x128.



**Figure 40:** Speed difference between the ego and lead vehicles across all time steps for a network architecture of 128x128.



## Chunk Size Ablation Study

Table 9 compares the performance of various chunk sizes across the 11 key metrics defined in Section 4. Every other hyperparameter is frozen as its default value, and the SAC algorithm is used. The best result for each metric is bolded.

**Table 9:** Chunk size performance comparison.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Chunk Size | 2 | 10 | 50 | 100 | 200 | 400 | 600 |
| Average Reward | -451.937 | -2.912 | -2.508 | -2.475 | -2.717 | -13.633 | **-2.321** |
| MAE (Ref. Speed) | 7.756 m/s | **1.886 m/s** | 1.926 m/s | 1.946 m/s | 1.967 m/s | 1.891 m/s | 1.988 m/s |
| RMSE (Ref. Speed) | 9.187 m/s | **2.354 m/s** | 2.401 m/s | 2.421 m/s | 2.439 m/s | 2.361 m/s | 2.485 m/s |
| Score (Ref. Speed) | -4.133 | **0.663** | 0.649 | 0.644 | 0.638 | 0.661 | 0.624 |
| Mean Distance to Lead | -36.912 m | 21.781 m | 18.580 m | 19.164 m | 14.806 m | 28.280 m | 17.206 m |
| Time in Safe Zone (%) | 17.6 % | 86.4 % | 93.8 % | 95.4 % | 95.9 % | 53.3 % | **97.5 %** |
| Min Recorded Distance | -180.843 m | **3.044 m** | 2.170 m | -0.802 m | -0.956 m | 2.793 m | 1.152 m |
| Max Recorded Distance | 70.283 m | 39.327 m | 34.924 m | 34.587 m | **30.252 m** | 53.730 m | 32.758 m |
| Mean Jerk | 0.001 m/s^2 | **-0.000 m/s^2** | -0.002 m/s^2 | **0.000 m/s^2** | -0.001 m/s^2 | -0.002 m/s^2 | 0.001 m/s^2 |
| Max Jerk | **2.246 m/s^3** | 3.008 m/s^3 | 3.898 m/s^3 | 3.733 m/s^3 | 3.539 m/s^3 | 3.817 m/s^3 | 3.520 m/s^3 |
| Jerk Variance | **0.257 m^2/s^6** | 0.341 m^2/s^6 | 2.559 m^2/s^6 | 1.562 m^2/s^6 | 2.580 m^2/s^6 | 3.109 m^2/s^6 | 2.157 m^2/s^6 |
| Mean Absolute Difference to Lead | 7.715 m/s | **1.805 m/s** | 1.882 m/s | 1.910 m/s | 1.935 m/s | 1.866 m/s | 1.944 m/s |

This ablation study explores how varying the episode chunk size impacts training performance and agent behavior. Chunk size determines how many timesteps are included per training episode, which directly affects the agent’s ability to observe and respond to temporal patterns. The results show a clear trend: both very small and very large chunk sizes can hinder performance, while moderate chunk sizes encourage balanced learning. Among all tested values, a chunk size of **100** yielded the most reliable and well-rounded behavior, despite not overly excelling in any category.

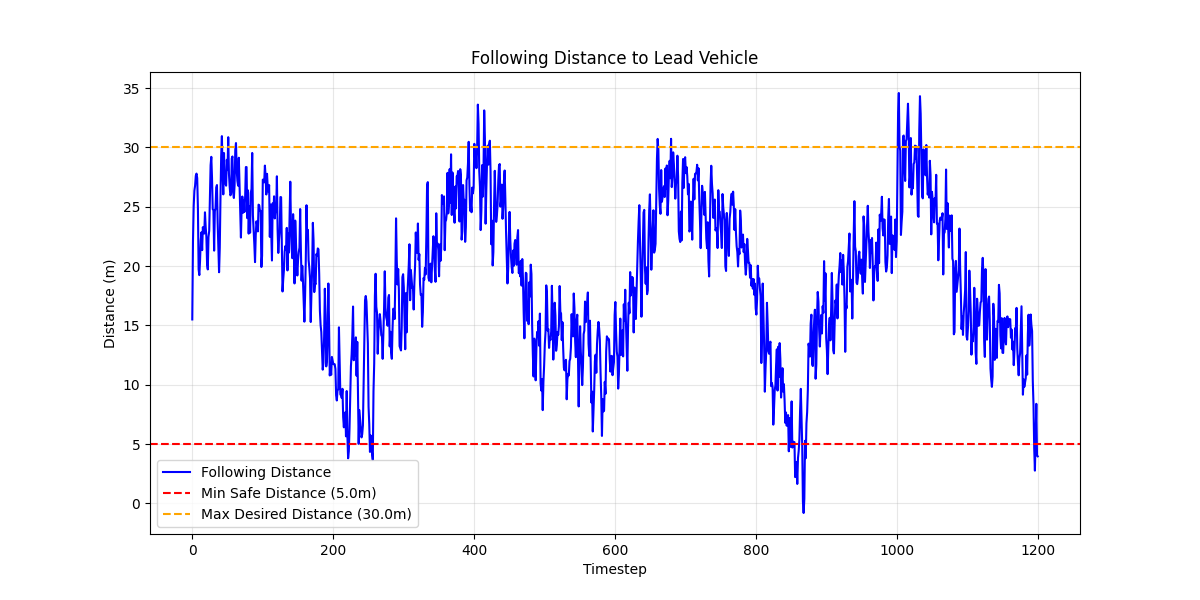
Extremely short chunks, such as size 2 and 10, led to unstable policies. The worst case was chunk size 2, which failed catastrophically across all metrics. It recorded a reward of -451.937, a massive mean speed error of 7.756 m/s, and a negative mean distance to the lead vehicle of -36.912 meters. This behavior is confirmed by the plots, which show the ego vehicle oscillating out of sync with the reference and lead speeds. Even chunk size 10, while performing better numerically, still displayed erratic distance tracking, frequent dips into unsafe zones, and a particularly awful learning curve as shown by the reward plot.

Mid-range chunk sizes performed much better. Among these, chunk size 100 stood out as the most consistent. It achieved the highest average reward of -2.475, while also maintaining a strong balance between speed tracking, distance control, and ride comfort. The agent stayed in the safe distance zone 95.4% of the time and maintained smooth, steady acceleration, as shown by its low jerk variance of 1.562. The distance plot showed very few violations of the 5–30-meter safe zone, and the reward curve converged the most smoothly of any configuration. Visually, the ego vehicle closely followed both the lead vehicle and the reference speed without large overcorrections.

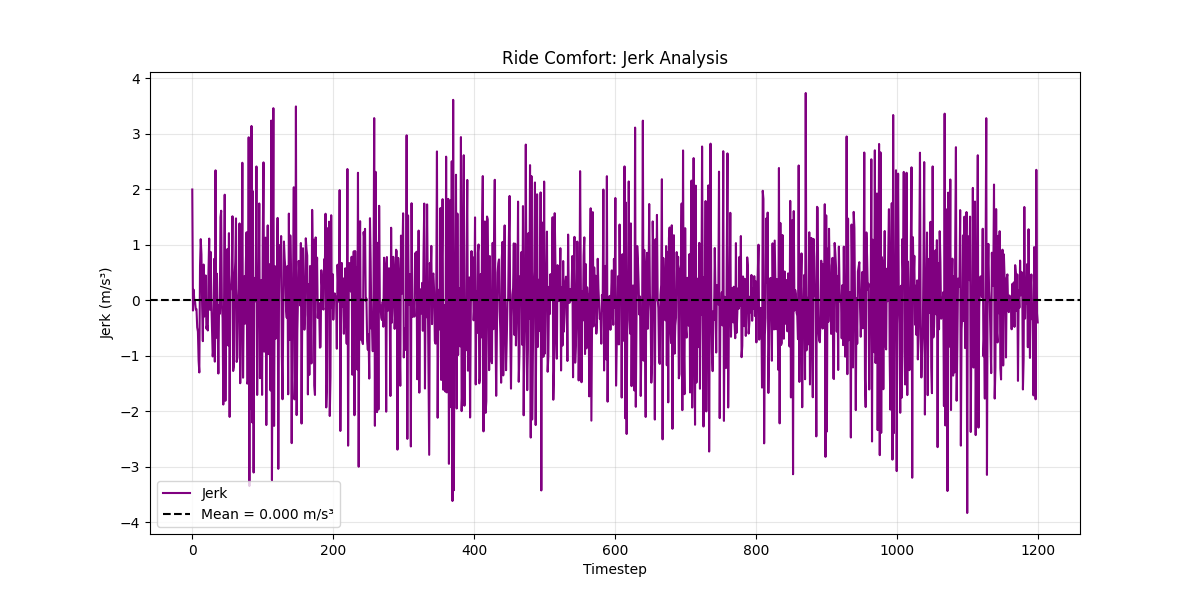
Larger chunk sizes such as 400 and 600 were also viable, with chunk size 600 achieving the highest safe zone time of 97.5%. However, these larger chunks came with trade-offs: their MAE was slightly worse, jerk variance was higher, and distance control was less consistent. It’s possible that training on overly long episodes caused the agent to overfit specific long-term trends, reducing its ability to adapt to local variations.

Considering all the results and visualizations, chunk size 100 is selected as the optimal configuration. It produces steady learning, consistent behavior, and robust generalization without sacrificing ride comfort or safety.

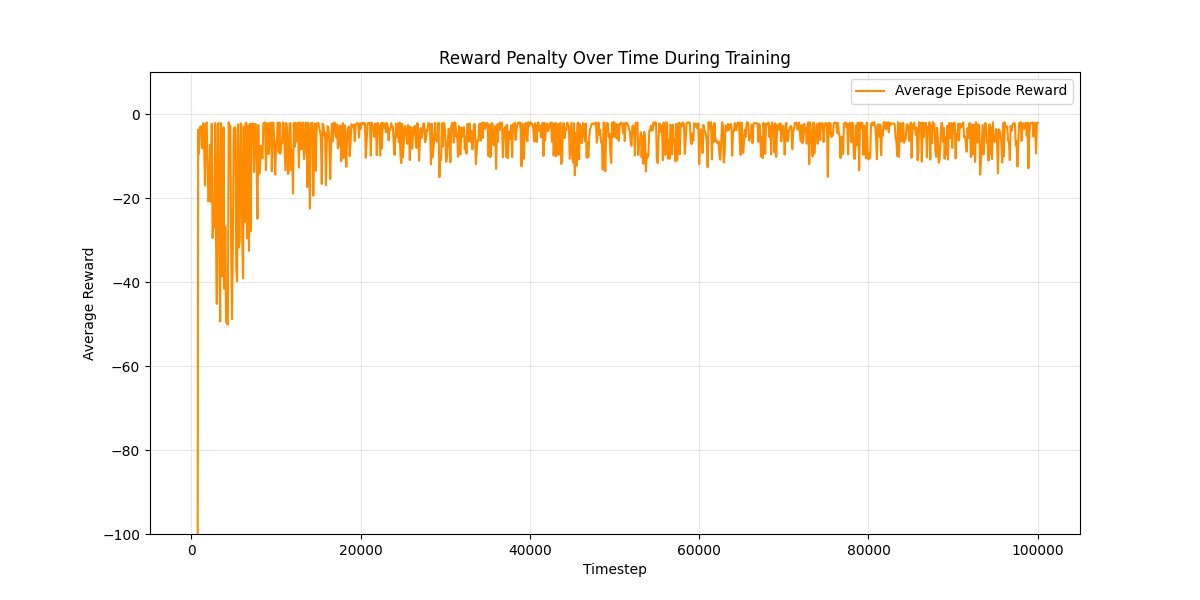
**Figure 41:** Following distance visualization across all time steps for a chunk size of 100.



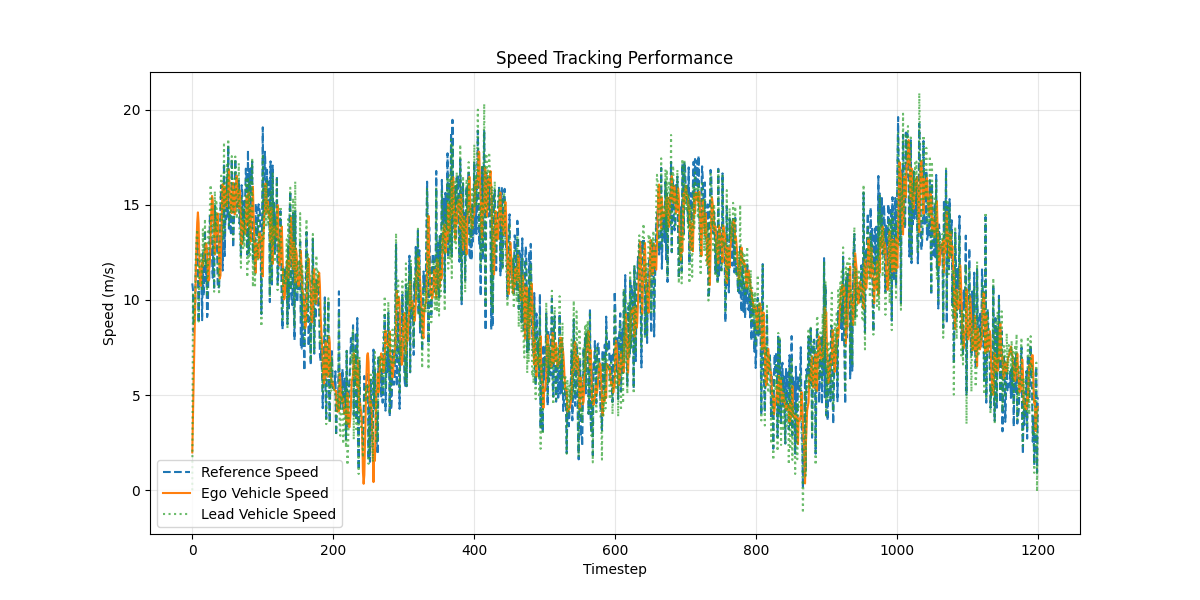
**Figure 42:** Jerk profile visualization across all time steps for a chunk size of 100.



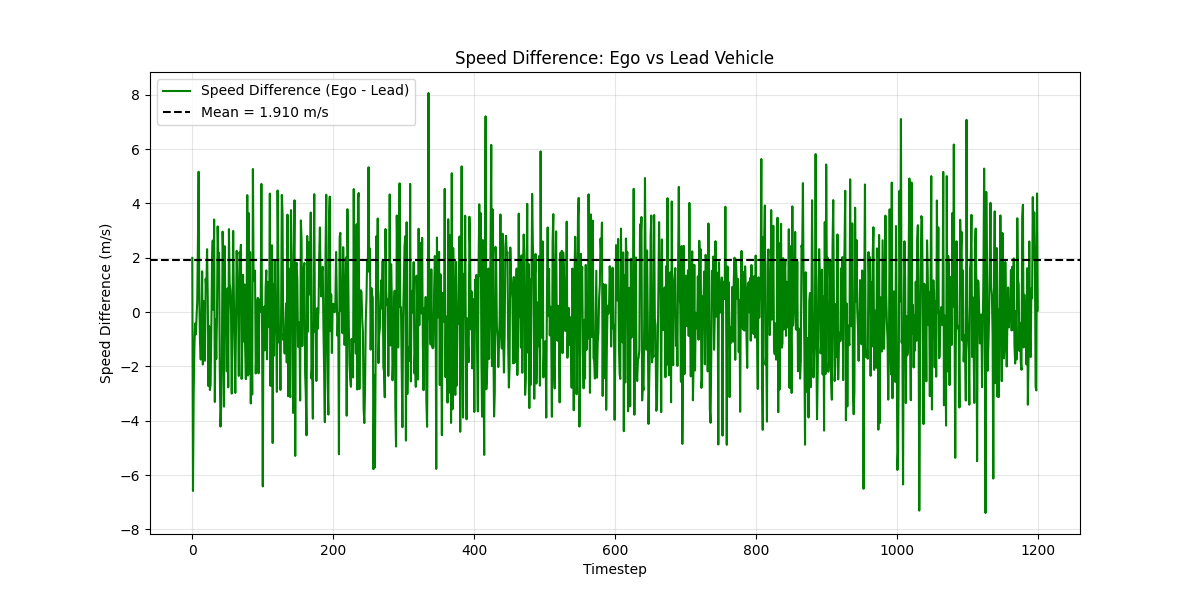
**Figure 43:** Reward penalty during training for a chunk size of 100.



**Figure 44:** Speed comparison for the ego, lead, and reference vehicles across all time steps for a chunk size of 100.



**Figure 45:** Speed difference between the ego and lead vehicles across all time steps for a chunk size of 100.



## Best Configurable Results Ablation Study

From the prior sections, the best hyperparameter configuration can be combined with the results below:

* Algorithm: SAC
* Batch Size: 256
* Buffer Size: 50,000
* Chunk Size: 100
* Entropy Coefficient: 0.05
* Gamma: 0.95
* Learning Rate: 0.003
* Network Architecture: 128x128
* Tau: 0.0001

Putting this configuration together and training a model using the same training environment, the resultant metrics are shown below in Table 10.

**Table 10:** Results using ideal configuration of hyperparameters.

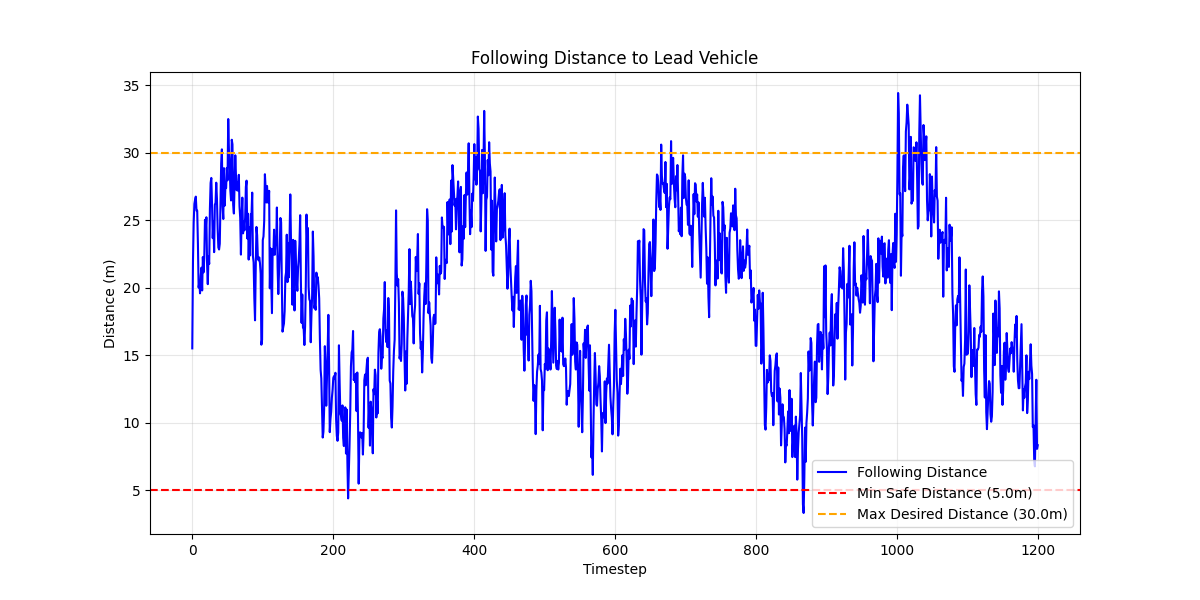
|  |  |
| --- | --- |
| Metric | Result |
| Average Reward | -2.274 |
| MAE (Ref. Speed) | 1.961 m/s |
| RMSE (Ref. Speed) | 2.437 m/s |
| Score (Ref. Speed) | 0.639 |
| Mean Distance to Lead | 19.297 m |
| Time in Safe Zone (%) | 97.0% |
| Min Recorded Distance | 3.306 m |
| Max Recorded Distance | 34.416 m |
| Mean Jerk | -0.001 m/s^2 |
| Max Jerk | 3.609 m/s^3 |
| Jerk Variance | 2.319 m^2/s^6 |
| Mean Absolute Difference to Lead | 1.901 m/s |

This experiment combined the SAC algorithm with a batch size of 256, buffer size of 50,000, chunk size of 100, entropy coefficient of 0.05, gamma value of 0.95, learning rate of 0.003, a 128x128 network architecture, and tau value of 0.0001. As shown in Table 10, this ideal configuration delivered strong overall performance, achieving an average reward of -2.274, which slightly outperformed most individual experiments while remaining close to the top performer in that metric (reward of -2.203 seen during the learning rate ablation). It maintained an MAE of 1.961 m/s and RMSE of 2.437 m/s in tracking the reference speed—values that were competitive, though marginally higher than the lowest errors seen during other studies, such as the 0.0007 learning rate or 0.9 gamma trials. Speed tracking was visually confirmed in Figure 49, which shows the ego vehicle staying well-aligned with both the reference and lead vehicle speeds. The score of 0.639 was slightly below the best achieved (0.685), indicating there may still be subtle compromises when multiple optimal values are combined.

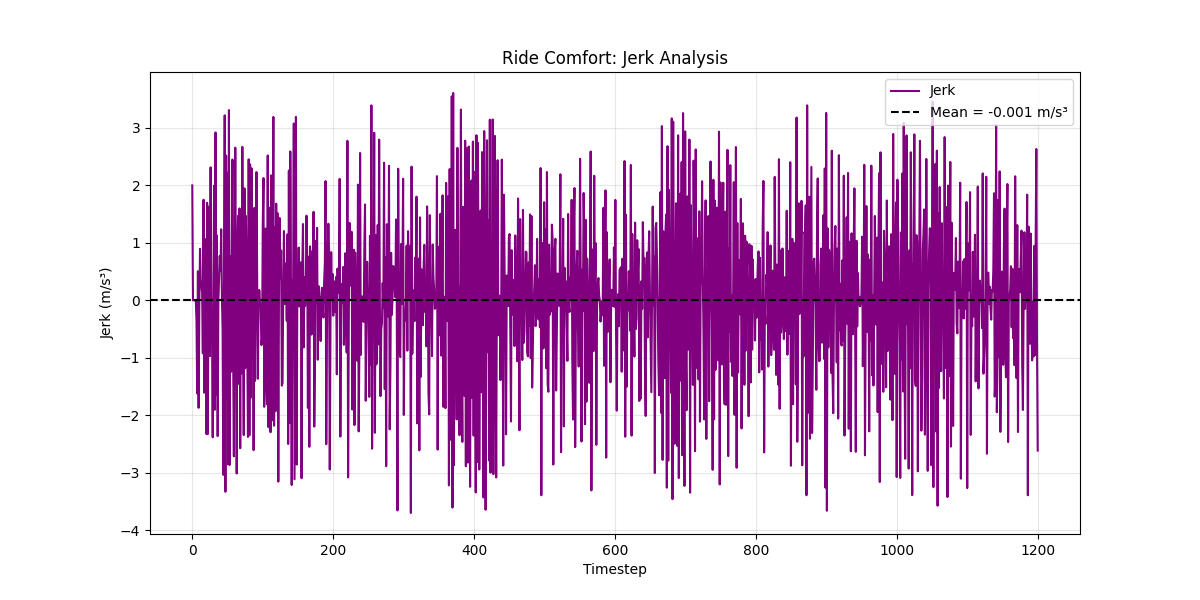
In terms of safety, the model performed reliably, with the ego vehicle staying within the 5 to 30-meter safe following distance 97.0% of the time. Although this did not surpass the 99.7% observed in the entropy coefficient experiment with a value of 0.05, it still represents one of the most consistent and robust safety outcomes across all tested configurations. Figure 46 supports this by showing a generally smooth and bounded following distance profile. The minimum and maximum recorded distances (3.306 m and 34.416 m, respectively) did not largely break the margins, avoiding the extreme outliers that emerged during other trials. Regarding comfort, the jerk profile in Figure 47 reveals a steady distribution of acceleration changes with no major spikes. The model recorded a mean jerk of -0.001 and a variance of 2.319, which, while not the lowest jerk variance seen in isolation, still suggests smooth enough transitions to be acceptable in practice. Compared to the entropy coefficient ablation, where the lowest jerk variance (1.562) was achieved at the cost of reduced safety, this configuration balanced both domains more evenly. The training reward trajectory (Figure 48) also exhibits rapid convergence (with minor fluctuations). The speed difference between the ego and lead vehicles, shown in Figure 50, confirms that the model reliably maintained a consistent pacing behavior, with a mean absolute difference of 1.901 m/s.

While the ideal configuration did not strictly dominate every single metric, it produced a consistently strong model that balanced tracking accuracy, safety adherence, ride comfort, and training stability. These results support the idea that optimizing hyperparameters in isolation can identify individual trends, but combining optimal settings can sometimes yield a synergistic effect that generalizes well across all target behaviors but also can compromise each individual setting’s benefits. However, given these tabular results, this configuration represents the most dependable real-world deployment candidate produced in this study.

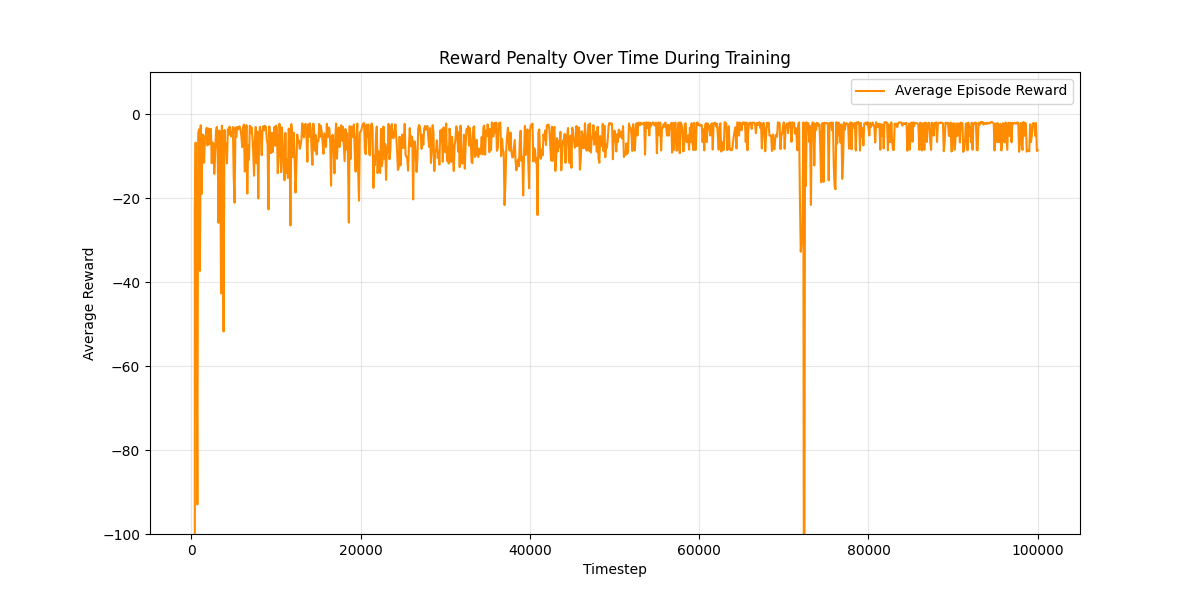
**Figure 46:** Following distance visualization across all time steps for the best hyperparameter configuration.



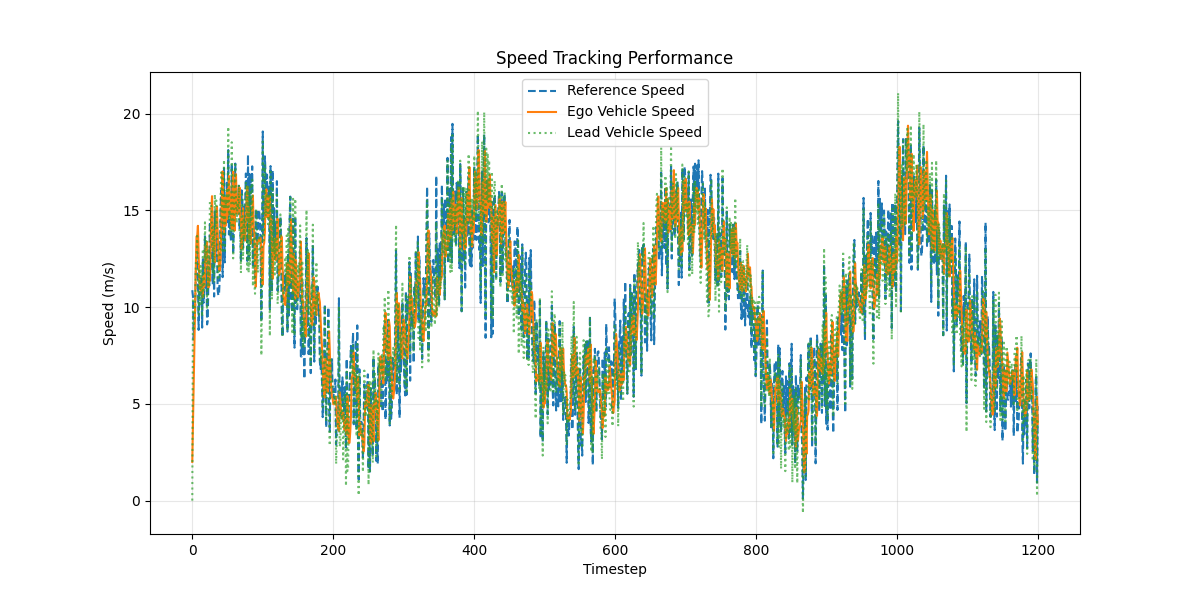
**Figure 47:** Jerk profile visualization across all time steps for the best hyperparameter configuration.



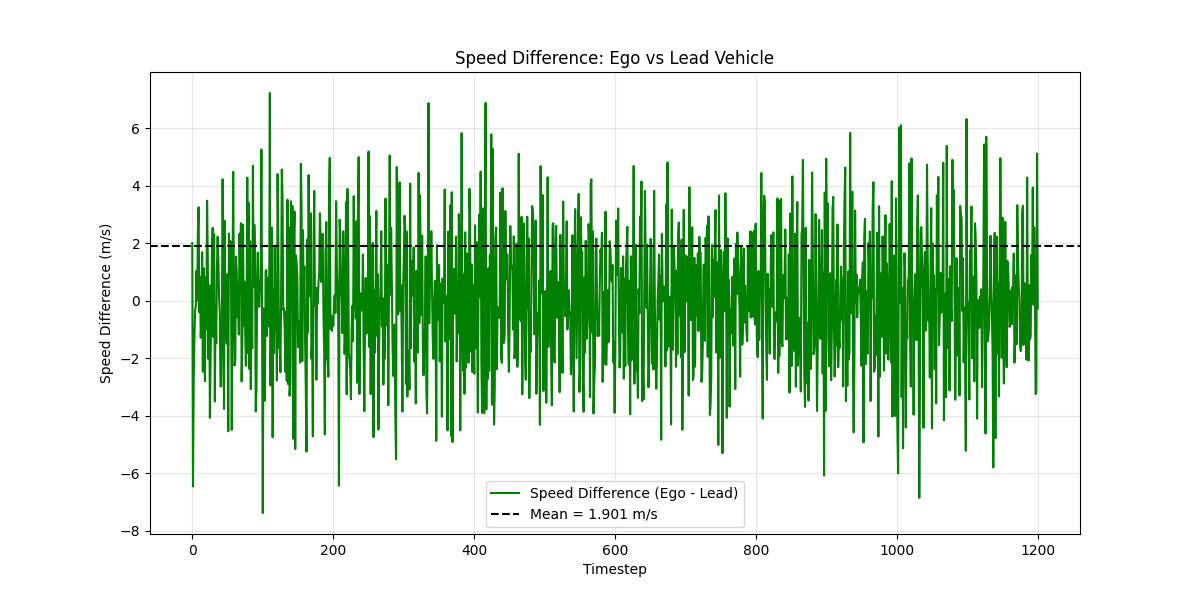
**Figure 48:** Reward penalty during training for the best hyperparameter configuration.



**Figure 49:** Speed comparison for the ego, lead, and reference vehicles across all time steps for the best hyperparameter configuration.



**Figure 50:** Speed difference between the ego and lead vehicles across all time steps for the best hyperparameter configuration.



# Discussion

The primary goal of this assignment was to extend a baseline reinforcement learning codebase to implement an Adaptive Cruise Control (ACC) system that maintains both a target reference speed and a safe following distance from a lead vehicle. This was achieved by applying extensive modifications to the environment, introducing a lead vehicle, adjusting the observation and action space, and redesigning the reward function to account for ride comfort, speed tracking, and safety.

To evaluate and improve the RL model, I systematically explored the effect of several hyperparameters using ablation studies. These included RL algorithms, learning rate, batch size, buffer size, tau, gamma, entropy coefficient, network architecture, and episode chunk size. Each hyperparameter was tested independently while all others remained fixed, allowing for a clear analysis of its isolated impact. Across nearly all metrics, the SAC algorithm consistently outperformed PPO, TD3, and DDPG, making it the preferred base model. Among the ablations, learning rate 0.003, batch size 256, buffer size 50,000, tau 0.0001, gamma 0.95, entropy coefficient 0.05, and a network architecture of 128×128 all demonstrated strong safety and stability performance, even when some were not optimal for tracking metrics.

To validate these results, a final experiment combined the best settings from each ablation into a single model configuration. While this “ideal” configuration did not achieve the best score in every metric, it consistently produced one of the most stable and reliable performances overall. Specifically, the model achieved an average reward of -2.274, and a time in safe zone of 97.0%. The model also tracked reference speed well, with a MAE of 1.961 m/s and RMSE of 2.437 m/s. Compared to individual ablations that prioritized single aspects (like gamma = 0.9 for best or entropy = 0.05 for best safety), this final model struck a practical balance between accuracy, safety, and comfort.

Despite these successes, there is still room for improvement. Certain experiments revealed trade-offs between speed tracking and safety, especially in models with overly aggressive learning rates or short episode lengths. Additionally, while the final jerk values were reasonable, some configurations produced higher variance than desirable. Future improvements could focus on adjusting the reward structure to better balance speed and safety, or trying other RL algorithms to see if they offer better training stability. Alternative architectures like RNNs could also help improve temporal consistency for long episodes given their association with time-series data.

Overall, this project successfully met the assignment goals by creating a robust ACC system that adapts to a dynamic lead vehicle, respects safety limits, and delivers smooth driving behavior. The model modifications, reward shaping, and visualizations collectively illustrate a comprehensive understanding of reinforcement learning in a safety-critical context.