# AIEA Report Fall 2025

Caleb Bergen

## Abstract

This project was inspired by real-world cases where autonomous vehicles (AVs) failed to successfully navigate unfamiliar environments and inadvertently made things worse. This project is also meant to be a continuation of the work done by William Self. To address these challenges, I created a framework using large language models (LLMs) to reason about what action should be taken given what is happening in the environment. LLMs were chosen because they are trained on a vast amount of human-generated data, so the LLMs should be able to pick up on what should and should not be done while driving. I used the models GPT-5-nano and GPT-4o for their mix of power and speed. The process behind getting the model to take an action involved taking the information given by the environment, refactoring and structuring it into a format GPT can understand, and finally prompting GPT to make a decision based on the current state of the environment.

## 1 Introduction

Autonomous vehicles provide clear advantages such as improved safety, reduced congestion and vehicle emissions, and greater accessibility [1]. Companies like Zoox and Waymo have been developing autonomous vehicle technology for years now, and they have been expanding access to these vehicles over time. While these technologies have significantly improved, they are still not perfect, and there have been serious accidents as a result of decisions made by these vehicles. One major example occurred in San Francisco in October of 2023. A pedestrian crossing the street was hit by a car being driven by a human, and the collision sent the pedestrian into the path of the Cruise vehicle. While the Cruise vehicle did initially stop, it then tried to pull over while the pedestrian was in front of the Cruise car and ended up dragging the pedestrian along the road for another 20 feet before ultimately stopping [2]. The pedestrian was injured even more severely as a result of the actions taken by the Cruise vehicle, and as a result of this accident, Cruise is no longer permitted to have autonomous vehicles driving in the state of California.

The accidents that do happen generally occur in situations that weren't originally included in the training data because they weren't thought of ahead of time. These are difficult to prepare autonomous vehicles for due to how they learn. Autonomous vehicles learn by being exposed to a variety of scenarios and identifying what constitutes a good action in those situations. They can then apply what has worked in the past to the current scenario they are faced with, but this isn't guaranteed to be effective. For many of the scenarios where autonomous vehicles cause or worsen car accidents, it is reasonable to say that a human would have been able to respond to the scenario more safely due to our more advanced reasoning capabilities. This is where large language models (LLMs) like ChatGPT can come in handy, because they have been trained on large quantities of human-generated information and retain some of the human-like reasoning that is present in what they are trained on. The goal of this project is leverage this reasoning to create a driving assistant that automatically takes the safest actions in driving environments.

## 2 Background

Using LLMs to power autonomous vehicles has become a much larger field of study over the last few years, as the greater reasoning capacity of LLMs compared to previous AV methods provides an opportunity to make autonomous driving even safer. One particular study discusses the efforts of many other researchers in what they call Large Language Models for Autonomous Driving (LLM4AD) [3]. The subtopics examined by the study separate LLM4AD into perception, planning and control, generation, question answering, and evaluation/benchmark.

The first subtopic addressed is perception, which is how the LLM interprets the problem given to it. The data given to the LLM can vary from numerical data to pixels from an image or video, to physical data from various sensors, such as LIDAR or cameras. If the data given to the LLM is not interpretable by it, it won't know how to make a proper decision, so its outputs will be useless. Therefore, it is important to ensure that properly formatted data is given to them. LLMs are especially good at picking up on trends while only being exposed to small sets of training data (few-shot learning) [4]. This is something that traditional AV implementations can struggle with, as their decision-making largely depends on what they have seen, and they can have difficulty reasoning when exposed to new information.

Planning and control involve the agent predicting what will happen in the environment around it and how that affects what it should do. Thinking through what could happen next is very important as a human when driving, thus it is necessary that any AV agent needs to do the same.

Generation involves LLMs using their vast knowledge bases to create intricate videos from singular prompts without needing any other information or time to train [5]. Videos can be created that detail any number of possible driving scenarios with little effort, which avoids needing to manually generate scenarios either physically or digitally. This helps to reduce the amount of time needed to train AVs, as the time it takes to generate training data for them is sped up, as well as make them more robust by making it easier to expose them to a wider variety of scenarios.

Question answering is used to see how the LLM understands and interprets the given environment. These responses are useful for determining if the LLM is thinking about problems in the expected manner.

Evaluating and creating benchmarks for LLMs performance is very important, as it creates a way to see if a LLM would be viable in a real car and it allows different LLMs performance to be compared to each other. One framework that exists for testing LLMs is called LangAuto (Language-guided Autonomous Driving) CARLA benchmark. The framework tests an agent's performance across 8 different towns and 16 different environmental conditions [6]. The agents are scored based on what percentage of a predetermined path they follow, as well as whether or not they made any infractions, such as a collision or traffic law violations.

All of the listed parts of LLM4AD are essential for the LLM to be able to make good decisions based on the environment around it. I have spent the most time working on perception and planning/control, and I intend to work more on question answering and benchmarking over time.

# 3 Methods

## 3.1 RL Agent

As previously stated, I used Deep Q-Network (DQN) to create a baseline for comparison due to its robustness and stable learning. To make the program, I took the DQN agent I made for the auditing program last quarter and modified it slightly to be able to interact with the new environment.

## 3.2 Environment

The environment I used was the HighwayEnv environment made by OpenAI. The environment by default is a four-lane road that has one user-controlled vehicle, called the ego vehicle, and many other vehicles that are controlled by the environment. Each episode ends either when the ego vehicle collides with another vehicle or after a certain number of timesteps have passed.

For GPT to be able to make decisions on what action to take, it needs to be able to interpret the environment. There are a couple of different ways that the environment returns the current state at each timestep, but both are numerical. One way is through the obs matrix that contains the x and y position as well as the x and y velocity for the ego vehicle and the four closest vehicles to it. However, I had some problems using these numbers specifically for the lane position. There were times when the calculated lane position did not match what was visibly shown in the environment. Through some research, I found that there is another way to get the information I need about the positions of the vehicles. The environment contains an object for the road, which contains a list of all of the vehicles on it. Each vehicle in this list has attributes for max speed, current position, and more. The ones that matter for what I am doing are the attributes for the current position, current speed, and what lane the vehicle is in.

The position for a vehicle is stored in a vector where the first element is the x position and the second element is the y position. The x position tracks how far along the road a vehicle is, and the y position tracks the vehicle's vertical placement along the road. The speed for a vehicle is stored as a float called speed. The possible values for speed are predefined as a range from -40 to 40 meters per second, with an initial value that is randomly set between 23 and 25 meters per second. The lane that a particular vehicle is in gets stored in the lane_index as a tuple of two strings and an int. The two strings are not important for determining the lane, but the int is important because it stores the lane a vehicle is currently in. The lanes are numbered 0 to 3 by default, so I added one to the lane number to make it follow a more logical numbering scheme of 1 to 4.

The list of vehicles returned by the environment does not necessarily put the ego vehicle as the first element, so I had to keep track of it separately and append it to the front of the list after all other vehicles are added. This ensures that GPT knows which set of information corresponds to the ego vehicle and which information is from the other vehicles. Another change I had to make was filtering out some of the vehicles on the road. This is because the list of vehicles from the environment contains every single vehicle on the road, including ones that are sometimes 1000+ meters away from the ego vehicle. There is no need to have all of those vehicles kept in the list given to GPT, as it would just take longer to compute an action given more values, so at each timestep, I exclude any vehicle that is more than 100 meters away from the ego vehicle in what is shown to GPT.

## 3.3 GPT-5-nano

I originally used GPT-5-nano as it is the fastest GPT-5 model. I wanted to ensure that the model I chose had the capabilities to reason well about what actions to take. After using this model, I got good results in its reasoning capabilities; however, it would take around 20 seconds on average to choose a decision for a single timestep. This is not very practical as a lot can change on a road in 20 seconds, so decisions would need to be made much faster than this. I realized that William used GPT-4o in his research for its speed, so I gave it a try to see how much that would change things. Switching to GPT-4o reduced response times from 20 seconds on average to about 5 seconds, which is a noticeable decrease but still needs to be faster for this to be practical.

One aspect I had problems with at first was getting a consistent numerical output of what action should be taken. Initially I prompted GPT to make the first character of its response be the action that should be taken. This proved to work most of the time, but sometimes the response wouldn't follow the requested pattern which would result in the program crashing. After some research I found that you can structure how the response is given from GPT, and this was able to fix the problem. I had the response that contained the reasoning for the choice of action taken given as a separate output from the numerical action to take. This ensured that the correct action was able to be identified separately from the response, and ensured the proper action would be taken.

## 4 Results

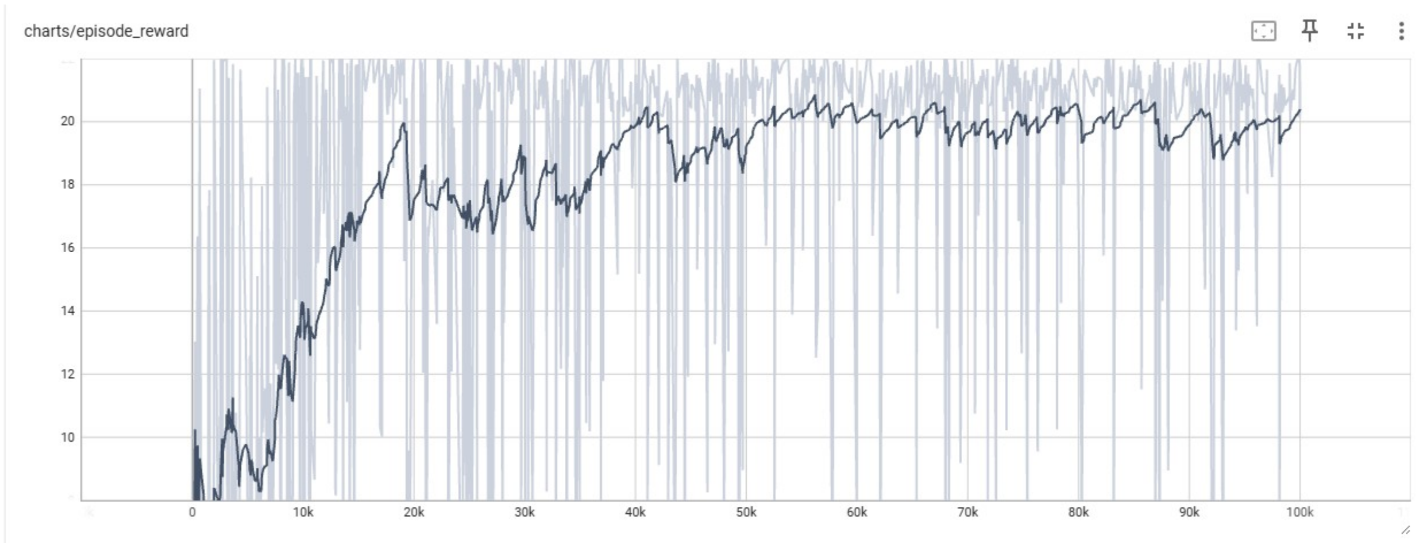

Figure 1: Reward curve showing average reward of DQN agent over 100,000 episodes.

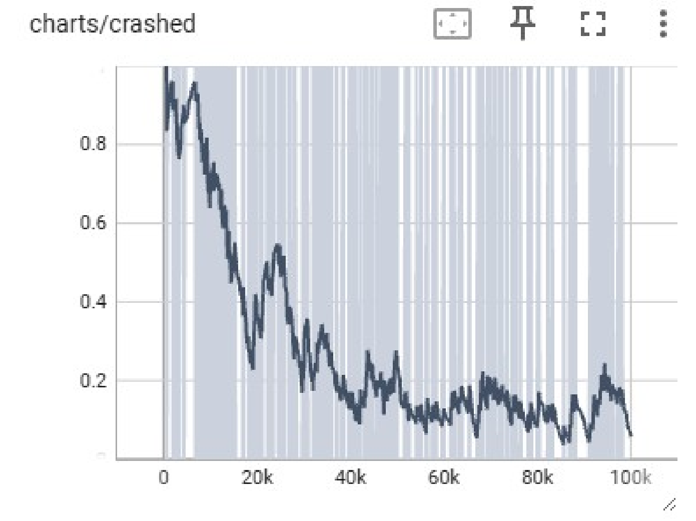

Figure 2: How often the DQN hits another vehicle.

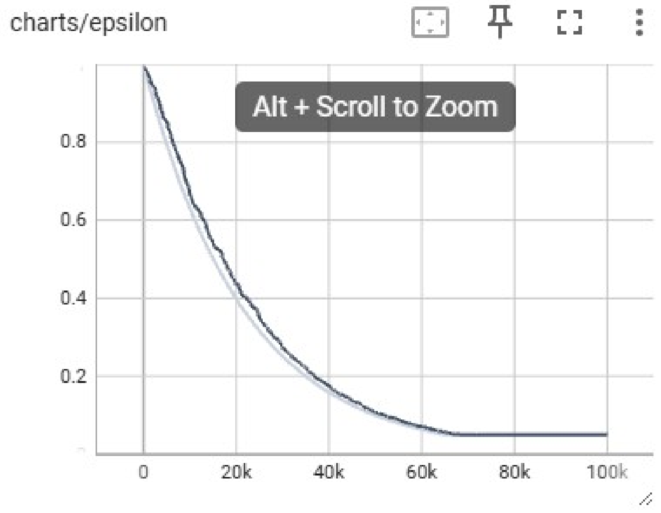

Figure 3: Decay of epsilon over time. Epsilon goes to a minimum of 0.05 to reflect the switch in the agent from exploration to exploitation.

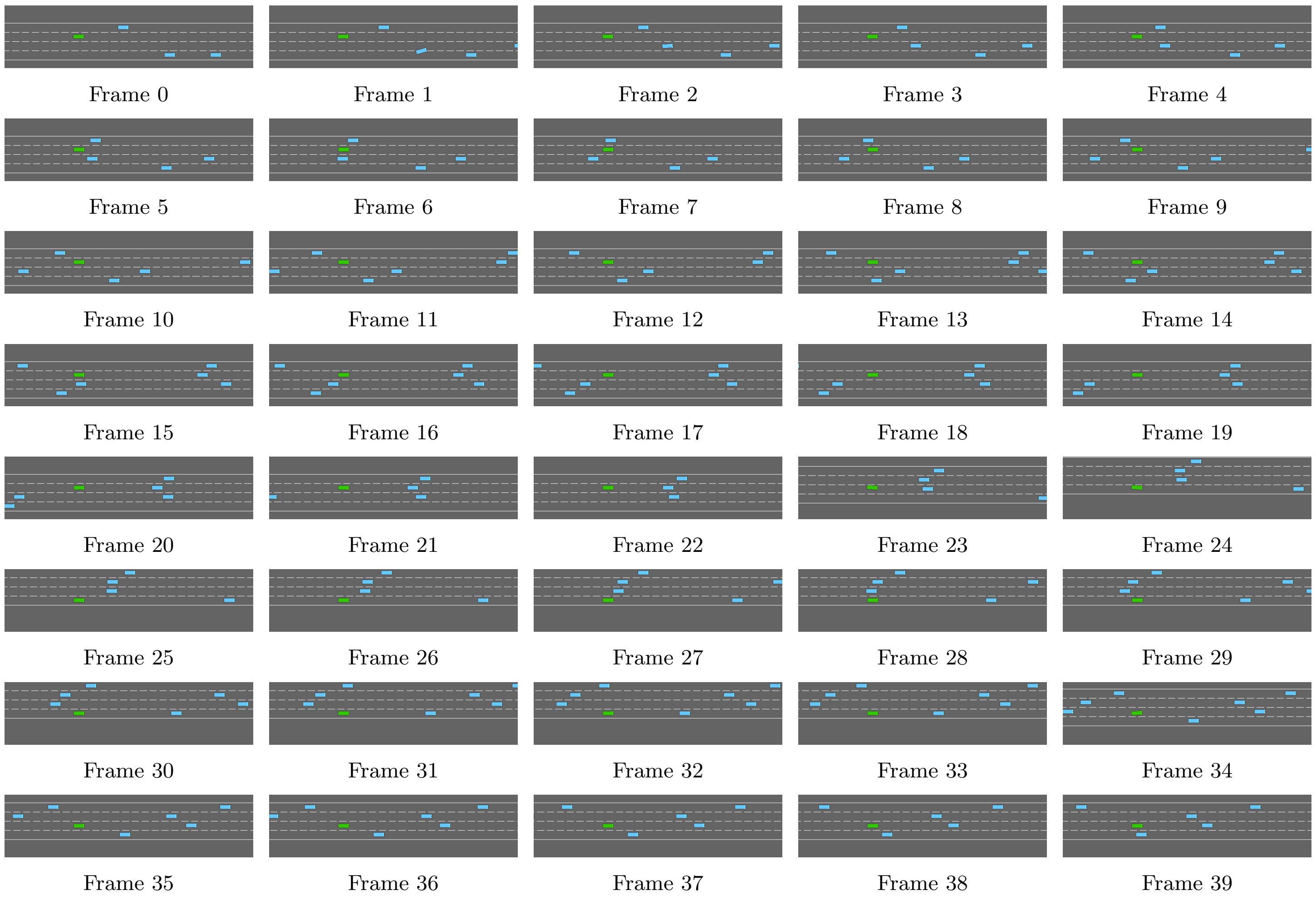

Figure 4: An example run of GPT controlling the ego vehicle

Figure 1 shows the average reward of the DQN agent over 100,000 episodes. Figure 2 shows how often the DQN agent crashed into another vehicle. After each episode, a 1 is recorded if the car crashed, and a 0 is recorded if it did not. So the value at any point corresponds to what percentage of the time the car is crashing across episodes. Figure 3 shows that epsilon, or the exploration rate, decays to 0.05 over time to ensure proper time for exploration, while still ending with the agent exploiting what it learned. When visually watching the agent after its training is completed, it is able to do a good job avoiding cars. However, the manner in which it does this isn't optimal for driving a vehicle. It tends to drive as fast as possible, a flaw of part of its reward being given for driving fast, and change lanes often to pass other vehicles to avoid crashing. These are not safe driving habits, so this method of controlling an AV would not be very beneficial to implement in an actual car.

The results of GPT have a clear contrast compared to the DQN agent. GPT tends to take a lot fewer actions than the DQN agent, and tries to prioritize staying in its current lane unless absolutely necessary. The most common action taken by GPT is idle, as it tries to stay at its current speed and flow with the other traffic on the road. From the output it gives as reasoning to why it chooses an action it is able to identify and avoid other vehicles on the road. Figure 4 shows an example run of what GPT does when controlling the ego vehicle. At the beginning of the run it safely passes a few cars, and then over the rest of the episode it settles near a few other cars.

## 5 Discussion and Conclusion

After comparing the performance of both the DQN agent and the GPT agent, it is clear that the GPT agent is ultimately the better option for controlling a vehicle. The DQN agent has shown an ability to drive quickly while generally avoiding crashing. However, this isn't the ideal driving behavior as it hard for other drivers to predict what the ego vehicle will do and can lead to more accidents. These behaviors are clearly improved upon by the GPT agent. GPT keeps the ego vehicle's speed lower as well as closer to the speed of the surrounding vehicles on the road. It also changes less often and tends to only do it when necessary to flow best with traffic. This helps it follow a much less erratic, more predictable behavior than the DQN agent.

That said, there are some limitations to this current approach. The most serious limitation to this approach is the occurrence of incorrect decisions by GPT. While GPT is able to correctly identify what is happening in the environment and make a good choice a good amount of the time, this doesn't always happen. There are cases where it seems not to realize, or just ignore, the fact that there is another car in an adjacent lane as it switches into that lane and crashes with the other car. There are other times when it will switch to another lane to try to flow better with traffic, but it switches into a lane where it is now even closer and faster than the car in front of it. I have a few thoughts on why these issues might be occurring. The first idea I had was that the way I am processing the data and feeding it to GPT is causing some problems in its understanding of the environment. I could mistakenly be doing things like mislabeling data or giving it data in an incorrect order. Another problem could be that the prompt given to GPT could be causing it to think about the problem in an incorrect way, and thus cause its results to deviate from the expected outcome. The last reason I think might be happening is due to hallucination from GPT. Hallucination in an LLM occurs when it tries to answer the question given to it with a probable response that it comes up with that ends up being incorrect [7]. Another issue is how long it takes to get a response from each time GPT is queried. If GPT is only asked to explain the choice it chose it will take 2-3 seconds, but if it is asked to explain why or not each action individual action should or shouldn't be taken it can take closer to 10 seconds. This is much too slow as conditions on a road can change in an instant, and sometimes split seconds need to be made.

All of these issues are things I plan on looking into and trying to address next quarter. Fixing these issues should make GPTs decisions more consistent and more in line with the safe driving practices that are desired to be followed.

## References

[1] Alliance for Automotive Innovation, *Benefits of automated vehicles (avs)*, https://www.autosinnovate.org/initiatives/innovation/autonomous-vehicles/benefits-of-havs, Accessed: 2025-12-05, 2025.

[2] Kevin Truong, *Cruise hid video of woman being dragged along san francisco street, dmv says*, https://sfstandard.com/2023/10/24/cruise-robotaxi-dmv-suspension-video/, Accessed: 2025-12-05, 2025.

[3] Z. Yang, X. Jia, H. Li, and J. Yan, "Llm4drive: A survey of large language models for autonomous driving," 2025. arXiv: arXiv:2311.01043 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2311.01043.

[4] J. J. P, K. Palanisamy, Y.-W. Chao, X. Du, and Y. Xiang, "Proto-clip: Vision-language prototypical network for few-shot learning," 2025. arXiv: arXiv:2307.03073 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2307.03073.

[5] L. Khachatryan et al., "Text2video-zero: Text-to-image diffusion models are zero-shot video generators," 2025. arXiv: arXiv:2303.13439v1 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2303.13439v1.

[6] H. Shao, Y. Hu, L. Wang, S. L. Waslander, Y. Liu, and H. Li, "Lmdrive: Closed-loop end-to-end driving with large language models," 2025. arXiv: arXiv:2312.07488 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2312.07488.

[7] S. Banerjee, A. Agarwal, and S. Singla, "Llms will always hallucinate, and we need to live with this," 2025. arXiv: arXiv:2409.05746v1 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2409.05746v1.