Self-Driving Car with JavaScript Neural Networks and Machine Learning

ALX - S.E PORJECT

—

KHALED MOHAMED FATHALLAH MOHAMED

ALX-S. E, AFRICA
ALEXANDRIA, EGYPT

Overview

- create a neural network using JavaScript with no libraries.
- make a self-driving car simulation by implementing every component step-by-step.
- implement the car driving mechanics, how to define the environment, how to simulate some sensors, how to detect collisions, and how to make the car control itself using a neural network.
- covers how artificial neural networks work, by comparing them with the real neural networks in our brain. how to implement a neural network and how to visualize it so we can see it in action.

Goals:

1. Neural Network Creation:

a. Develop a neural network using JavaScript without relying on external libraries.
b. Understand how artificial neural networks work, comparing them with real neural networks in the brain.

2. Self-Driving Car Simulation:

a. Implement the car driving mechanics.
b. Define the environment for the simulation.
c. Simulate sensors for the car.
d. Implement collision detection.
e. Enable the car to control itself using the neural network.

Specifications:

Learning Approach:

- Encourage students to understand that they can achieve anything in modern computers with patience and any tool.
- Address the fear of trying new things and doing something different.

Project Challenge:

- Emphasize that creating a neural network without libraries and not using Python is challenging and fun.
- Highlight that the project aims to help individuals understand that they can start a career in data science with a simple introduction to neural networks.

Challenge:

THE PROJECT IS INTENDED TO **MAKE STUDENT UNDERSTAND THAT** WITH ANY TOOL AND PACIENT YOU CAD DO ANY THING IN MODERN COMPUTERS.
THE PROJECT WILL NOT SOLVE , **THE PROBLEM OF FEARING NEW THING** , AND FEAR OF CHANGE TO DO SOMETHING DIFFERENT THAN THE OTHERS DO
AND FROM THE OTHER SIDE , MAKING NEURAL NETWORK WITHOUT LIBRARY , AND WITHOUT USING PYTHON IS MORE CHALLENGE AND FUN 🙂
THE PROJECT WILL HELP **YOU TO UNDERSTAND THAT YOU CAN WITHOUT ANY NEW TOOLS** OR LIBRARY START YOUR GAME AND GET IN DATA-SCIENCE CAREER WITH SIMPLE START IN NEURAL NETWORK AND ITS IMPLEMENTATION

Risks:

Failure Risk:

- **Acknowledge the risk of not finishing the project in time or at all.**
- **Understand the potential impact on the team and their decisions for future projects.**
- **Emphasize the importance of trying hard and learning from the process.**

Existing Solutions:

Future Use of ANN:

- **If successful, the goal is to create a JavaScript neural network tool (ANN) that can be implemented in various websites for input anticipation.**
-

Technologies:

I. **Primary Language: JavaScript.**
II. **Learning Resources: Online and YouTube courses.**
III. **Background Knowledge: JavaScript, algebra, physics, and coding problem-solving.**
IV. **Development Environment: Visual Studio Code (VS-Code).**

Infrastructure:

TEAM:

- **Team Structure:**
  - **Team Leader: Khaled Mohamed FATHALLAH Mohamed.**
  - **Roles: Project Manager, Developer, Tester, Presenter.**
  -

Architecture Diagram for the Self-Driving Car Simulation MVP:

Neural Network Module:

- **Label: Neural Network**
- **Description: Represents the JavaScript-based neural network responsible for decision-making in the self-driving car.**

Simulation Environment:

- **Label: Simulation Environment**
- **Description: Defines the virtual environment in which the self-driving car operates. Includes roads, obstacles, and other relevant features.**

Car Mechanics:

- **Label: Car Mechanics**
- **Description: Implements the driving mechanics of the car, including acceleration, steering, and braking.**

Sensor Simulation:

- **Label: Sensor Simulation**
- **Description: Simulates the sensors on the car, such as cameras or LiDAR, providing input to the neural network.**

Collision Detection:

- **Label: Collision Detection**
- **Description: Monitors the environment for potential collisions and communicates with the neural network for decision-making.**

Control Module:

- **Label: Control Module**
- **Description: Integrates outputs from the neural network to control the car's movements and actions.**

User Interface (UI):

- **Label: User Interface**
- **Description: Optional graphical interface for users to observe the simulation and interact with the system.**

Tools to Create Architecture Diagram:

Lucid chart: An online diagramming tool that allows collaborative creation of flowcharts, diagrams, and more.
Draw.io: A free and open-source diagramming tool that can be used online or offline. It offers various shapes and elements for creating architecture diagrams.
Gliffy: A diagramming tool that integrates with Confluence and Jira, suitable for creating professional-looking architecture diagrams.
Microsoft Visio: A widely used diagramming tool that provides a range of templates for different types of diagrams, including system architectures

❖ Data Model Diagram for Self-Driving Car Simulation:

Entities:

- **Car:**
  - **Attributes: carID (Primary Key), positionX, positionY, speed, direction, etc.**
- **Environment:**
  - **Attributes: roadID (Primary Key), obstacleID, roadType, obstacleType, etc.**
- **NeuralNetworkInput:**
  - **Attributes: inputID (Primary Key), sensorData1, sensorData2, ..., sensorDataN.**
- **NeuralNetworkOutput:**
  - **Attributes: outputID (Primary Key), steeringAngle, acceleration, braking, etc.**

Relationships:

- **Car - Environment:**
  - **Relationship: Car is on Environment.**
  - **Cardinality: One-to-Many (A car can be on one type of road, but a road can have multiple cars).**
- **Car - NeuralNetworkInput:**
  - **Relationship: Car has NeuralNetworkInput.**
  - **Cardinality: One-to-One (Each car has a corresponding set of input data).**
- **NeuralNetworkOutput - Car:**
  - **Relationship: NeuralNetworkOutput controls Car.**

- **Cardinality: One-to-One (Each car has a corresponding set of output data).**

❖ Using SqlDBM:

Create Entities:

- **Open SqlDBM and create entities for Car, Environment, NeuralNetworkInput, and NeuralNetworkOutput.**
- **Add attributes to each entity.**

Define Relationships:

- **Establish relationships between entities using primary and foreign keys.**
- **Specify the cardinality of each relationship.**

Generate SQL Script:

- **SqlDBM allows you to generate SQL scripts based on your data model.**

Visualize the Model:

- **Utilize the visualization tools in SqlDBM to view and verify your data model.**

❖ User Stories and Pitfalls:

Research on User Stories:

- **User Stories are concise, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer.**
- **They typically follow the template: "As a [type of user], I want [an action] so that [benefit/value]."**
-

Pitfalls of General User Stories:

Vagueness: Lack of specificity can lead to misinterpretation and confusion.
Assumptions: Assuming everyone understands the context or goals can lead to misalignment.
Overly Technical Language: Using technical terms that users may not understand can create a communication barrier.
Too Much Detail: Providing excessive details can make the story lengthy and challenging to comprehend.

❖ User Stories for Self-Driving Car Simulation MVP:

User Story: Simulating Basic Driving Experience

- **As a simulation user, I want the self-driving car to navigate a predefined environment, so I can observe the basic driving mechanics in action and understand how the car responds to its surroundings.**

User Story: Implementing Collision Detection

- **As a simulation user, I want the self-driving car to detect obstacles and avoid collisions, so I can evaluate the safety and effectiveness of the collision detection mechanism in the simulation.**

User Story: Training Neural Network with Sensor Data

- **As a simulation user, I want the self-driving car to be trained using simulated sensor data, so I can observe how the neural network adapts and learns from the environment to make informed driving decisions.**

User Story: User Interface for Simulation Monitoring

- **As a simulation user, I want a user interface to monitor the self-driving car simulation in real-time, so I can observe the car's behavior, sensor inputs, and neural network outputs during the simulation.**

User Story: Tweaking Neural Network Parameters

- **As a simulation user, I want the ability to adjust parameters of the neural network, such as learning rate or layer architecture, so I can experiment and understand the impact of these changes on the car's driving behavior.**

Challenges:

1. Neural Network Implementation Complexity:

- **Discovery: The intricacies of implementing a neural network from scratch in JavaScript without relying on libraries proved to be more challenging than anticipated.**
- **Adaptation: The initial plan involved a straightforward neural network implementation, but due to the complexity, we had to break down the implementation into smaller, manageable steps. Additional research and learning resources were incorporated to address specific challenges encountered during the development process.**

2. Realistic Simulation Environment:

- **Discovery: Creating a realistic simulation environment that accurately reflects real-world driving scenarios posed unforeseen challenges. Ensuring the simulation reflects diverse driving conditions and scenarios was more complex than initially thought.**
- **Adaptation: The plan was adjusted to incorporate a more modular and extensible environment, allowing for the gradual addition of diverse scenarios. Collaboration with domain experts and further research into real-world driving data were integrated to enhance the realism of the simulation.**

3. User Interface Design and Usability:

- **Discovery: Crafting a user-friendly interface for monitoring the simulation and adjusting neural network parameters presented challenges in terms of design and usability.**
- **Adaptation: The plan was modified to allocate additional time and resources for UI/UX design considerations. Feedback loops with potential end-users were established to gather insights on improving the interface's intuitiveness and effectiveness.**

Project Updates:

1. Adjustment in Neural Network Implementation:

- Change: The initial plan aimed for a comprehensive neural network implementation from scratch. However, due to the complexity encountered during development, we've decided to integrate a lightweight JavaScript neural network library for better efficiency and performance.
- Reasoning: This adjustment is made to ensure a more robust and efficient implementation of the neural network. Leveraging a specialized library allows the project to focus on the unique aspects of the self-driving car simulation, promoting faster development and better maintainability.

2. Enhancements in Realistic Simulation Environment:

- Change: The plan initially included a modular environment, but we realized the need for more realistic driving scenarios. We're allocating additional time to research and incorporate real-world driving data for a more authentic simulation.
- Reasoning: To provide users with a simulation environment that closely mirrors real-world driving conditions, integrating authentic data is crucial. This change aims to enhance the overall realism and effectiveness of the self-driving car simulation.

3. Iterative User Interface Design:

- Change: The original plan allocated a fixed time for user interface design. However, we've decided to adopt an iterative approach, continuously refining the interface based on user feedback and usability testing.
- Reasoning: Recognizing the importance of a user-friendly interface, this change allows for continuous improvement based on user insights. Regular feedback loops will contribute to a more intuitive and effective user interface.

Progress:

WEEK 1 PROGRESS :

Rating: 7 out of 10

Measurement of Progress:

1. Development Milestones: Completed neural network integration and basic simulation environment setup.
2. User Stories: Two out of five user stories have been implemented successfully.
3. Collaboration: Improved team collaboration and defined clearer roles and responsibilities.
4. Learning Curve: Acknowledged and addressed challenges associated with the learning curve for specific technologies.

Reasoning:

1. **Positive Developments:**
- Successful integration of a lightweight JavaScript neural network library significantly advanced the project's core functionality.
- Initial simulation environment setup is complete, providing a foundation for further enhancements.

2. **Challenges and Adjustments:**
- The learning curve associated with certain aspects of the project led to adjustments in the original plan, but the team has adapted well.
- Iterative development allowed for addressing challenges promptly, fostering a positive learning environment.

3. **Assessment of Project Completion:**
- The adjusted timeline, iterative development approach, and continuous learning strategies are likely to contribute positively to project completion.
- The current progress indicates that the project is on track to meet the specified milestones. However, close monitoring and regular reassessment will be crucial to adapting to any unforeseen challenges that may arise.

Next Steps:

- **Prioritize remaining user stories based on their impact on the MVP.**
- **Implement more realistic scenarios in the simulation environment.**
- **Continue refining the user interface based on iterative feedback.**
- **Schedule additional knowledge-sharing sessions to address learning challenges.**

This rating reflects positive progress, the team's adaptability, and a proactive approach to overcoming challenges. Continuous monitoring and proactive adjustments will be maintained to ensure a successful and timely project completion.

WEEKS 2 PROGRESS :

Rating: 6 out of 10

Measurement of Progress:

1. Development Milestones: THIS WEEK I STARTED to understand neural network the math behind it and how it works , and how to implement it, in code.
2. User Stories: Now I have a general idea about how the user will interact with the system.
3. Collaboration: Some questions were asked on the internet and get the answers about them .
4. Learning Curve: it's hard and takes a long long time to finish.

Reasoning:

Positive Developments:

1. Milestone Completion:

- NOW i have repo and README.md , yes empty but i'm still learning the math behind the engineering of the project.
- User Story Progress: not so much for now
- Collaboration Improvements: Elaborate on any enhancements in team collaboration and the definition of roles.

2. Challenges and Adjustments:

- Learning Curve Adaptation: THE math , linear algebra and so on is the most important thing to do right now.
- **Technical Challenge:**
  - During the second week, the most challenging technical aspect was the implementation complexity associated with integrating the lightweight JavaScript neural network library. Initially, the plan aimed for a comprehensive neural network implementation from scratch, but the intricacies surpassed our expectations. The library integration was necessitated to enhance efficiency and performance, yet the adaptation process posed substantial hurdles.
  - The library's documentation was intricate, requiring an in-depth understanding of its functionalities and parameters. The challenge escalated as we sought to align the library's capabilities with our project's specific requirements. Debugging and troubleshooting became intricate tasks, demanding a meticulous approach to identify and rectify errors.
  - The learning curve associated with the library's integration proved steeper than anticipated, necessitating additional research and collaborative problem-solving sessions. While the decision to leverage an existing library aimed to streamline development, the process underscored the complexity inherent in integrating external tools within a bespoke project framework.
- **Non-Technical Challenge:**
  - During the second week, a significant non-technical challenge emerged: managing time effectively amid mid-term exams in the computer science pre-master's program. The collision of project commitments with academic responsibilities posed a formidable obstacle to the team's productivity.
  - The pre-master's mid-term exams demanded focused attention and thorough preparation, diverting valuable time and mental energy away from the project. Balancing the academic workload with the project's demands became a delicate act that required strategic planning and prioritization.
- Iterative Development: Continue to implement the neural network as functions and what input should be in and what output should be com from.

4. Assessment of Project Completion:

- Reflect on how the adjusted timeline, iterative development, and learning strategies influenced project completion during these weeks.
- Provide insights into the overall trajectory of the project and whether it is still on track based on the initial plan.

- Highlight any lessons learned or adjustments that might impact the remainder of the project.

Next Steps:

- **User Stories Prioritization: not until finish the big picture of how the final product will get out we can see then how the final user stories will be.**
- **Simulation Environment Enhancement: the neural network implementation is the most important thing now , then how to connect more than one together to achieve the final product.**
- **User Interface Refinement: still in consideration after the half of the project development.**
- **Knowledge-Sharing : not so much to do so now.**
- **Closing Thoughts:**

Summarize the rating : over all some is what is expected some is not as expected , even though it was in the consideration while making the plan for the project.