AI-Powered Precision: Revolutionizing Lunar Landings with Deep Learning and Reinforcement Learning

By: Saaketh Suvarna

Abstract: This paper investigates how a deep learning-based neural network can effectively enable precise and safe autonomous navigation for space vehicles in various environmental conditions. With a recent increase in interest in developing a lunar presence via the Artemis missions, the need for safe autonomous landing systems also increases. Using a convolutional neural network trained on topographic maps of a simulated landing zone for a robotic lander, the neural network is capable of identifying global regions of interest for flat landing spots along with local solutions to simulate real-time response during a mission scenario. Via simulated results with various initial landing trajectories, the neural network was capable of identifying if the initial landing spot can be deemed safe and if not, able to identify nearby landing locations that are attainable in terms of velocity limitations. The model efficiently calculates the best landing spots in the quickest time

Introduction

Throughout the 20th century, the pace of human technological advancement is truly remarkable, unyielding our pursuit of progress. Within a mere 66 years, humanity progressed through 2 monumental feats: The Wright Brothers' first flight in 1903, all the way to the landing of the first man on the moon in 1969. The growth of technology during this era was substantial. But since then, advancements in lunar exploration have plateaued.

So why haven't we returned to the moon? Despite the challenges posed by political and budgetary constraints, the moon, with its 4.5 billion-year history, presents a significantly hazardous environment for human exploration and habitation With its surface littered with craters and boulders, the chance of a safe landing is still a daunting task to overcome. The Apollo 11 Mission and its crew steered its way out of a potentially disastrous event for the US. At only 1600 feet above the moon's surface, Neil Armstrong's quick thinking saved the crew as he grabbed hold of the control and steered the Apollo vehicle away. The spacecraft's poor computer system had guided the crew straight into a boulder-strewn field, cutting away precious mission time as fuel sources were getting low. It's still a miracle today that the astronauts were able to touch down safely upon the Moon's surface.

Although it's been a while since we humans have touched the moon, many unmanned attempts have taken place, all facing the same challenge: landing safely. India's recent attempt at another lunar landing ended fatally. The Chandrayaan-2 Mission failed in September 2019 when the Vikram Lander crashed into the Moon's surface. Another major lunar crash took place the same year, as Israel's Beresheet lander came into a hard landing with the surface, cutting the connection with mission control and leaving a heartbroken impact on Israel's Aerospace Industries.

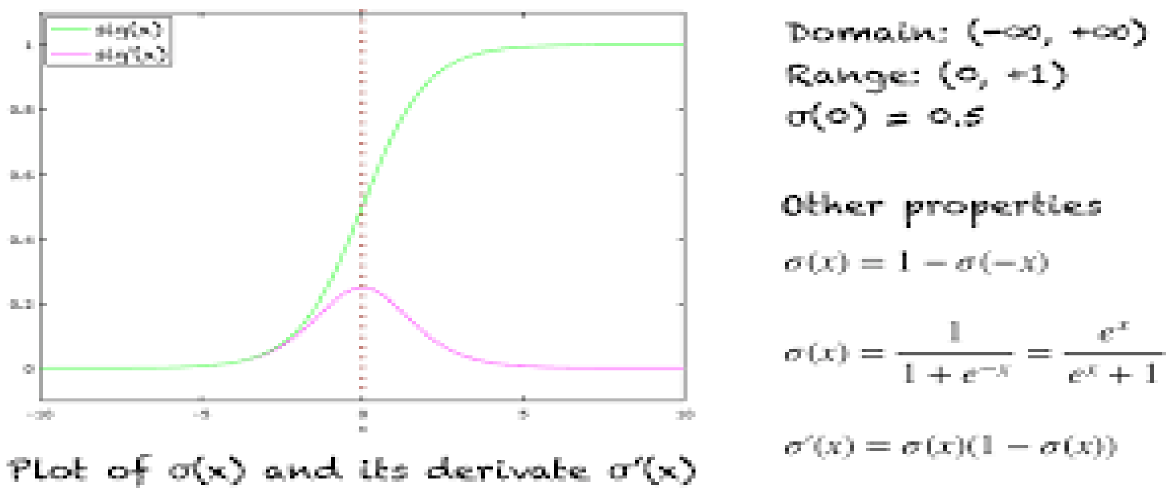
The primary objective of this project was to propose a novel approach to addressing these challenges in lunar landing. That approach involves the use of Artificial Intelligence. AI is a technological system involving the replication and enhancement of the traditional human cognitive process. It offers a new solution that surpasses human limitations in processing vast amounts of data and making real-time decisions. By leveraging AI's capability of environmental data interpretation, this research reveals a developed model that can autonomously detect safe landing spots on the moon from elevation data. The application of this system in the case of a real mission drastically reduces the common risks associated with lunar landings and graciously allows those to focus more on the content of the mission rather than landing.

Methods

Neural Networks

The heart of this project lies within a neural network, built in Python. A neural network is a method in AI that teaches computers to process data in a way that is inspired by the human brain. It uses interconnected neurons in a layered structure that resembles the human brain, creating an adaptive environment in which the computer learns from its mistakes and continues to improve [1]. They work by having layers within it that compute mathematical operations in each layer and feed the new data forward to the next layer. An example of a series of layers, is the input layer, a hidden layer that processes data further, and an output layer. Of course, the Neural Network can utilize as many layers as the designer creates. The project started with replicating a basic Neural Network in Python. This Neural Network starts with random weights and bias vectors, makes a prediction, and checks how close it is, finally repeating over and over until it gets it correct. To determine how close the vectors are, the dot product was used in one of the layers, as the dot product gives us how similar 2 vectors are in direction and magnitude. Another layer utilized was a function known as the Sigmoid Function, which is going to be very useful later on in this project.

Figure 1: Sigmoid Function Graph [1]



This is the graph of the sigmoid function, and it is considered a squashing function, as its range is only (0,1). The Sigmoid Function is a non-linear activation function that outputs within the range 0 and 1, which is useful for binary classification. The Final function used in one of the last layers of the Neural Network was the MSE (Mean Squared Error,) which computes the difference between the prediction and the actual value. Since the Mean Squared Error (MSE) involves squaring the errors, it invariably results in a positive value. When performed, the neural network starts to train itself and work through the different problem sets provided, finally returning a training graph for analysis.

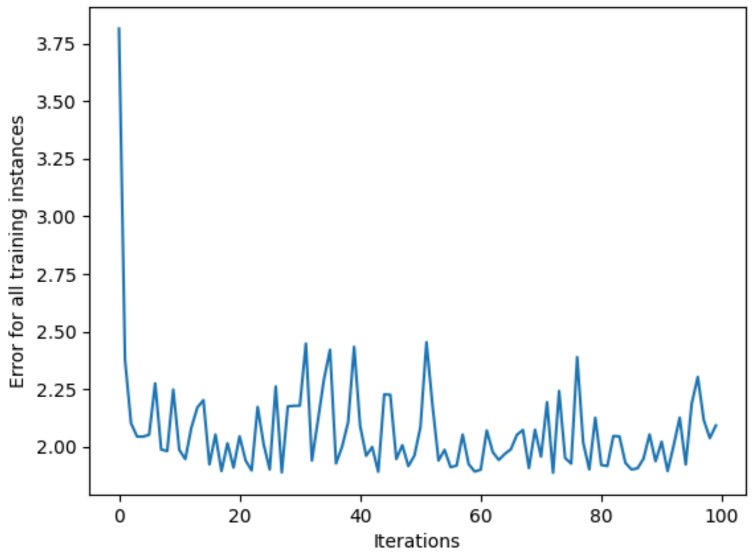


Figure 2: Training Error Results

After a thorough review of the existing work, the decision was made to advance the Neural Network. The enhancement enables the model to detect adjacent matching values within the matrices. This is used for elevation maps, as elevation map data is stored as matrices, with numbers representing elevation, and numbers that are similar and next to each other representing flat ground for the vehicle to land on. The approach used was modified in the choice of module. Previously using NumPy, Pytorch was employed due to its enhanced capabilities for neural network training. Specifically, PyTorch offers a more intuitive interface for handling tensors and has numerous built-in functionalities tailored for deep learning. This new neural network starts by importing Pytorch and giving it a set of random values in a matrix as an example of elevation data. This Neural Network includes 6 layers to process the matrix data, also using the Sigmoid Function as the last layer so it returns a definite 0 or 1 value if the matrices are similar or not. The model is trained by running it through thousands of iterations, in which it will optimize for better predictions each iteration. Ultimately, it gives us a 0 or 1 value, marking flat spots.

This Neural Network model faces challenges in customization for specific missions due to its lack of image recognition capabilities, which are crucial for accurately identifying flat spots. While the model can detect flat areas, its application in real mission scenarios is limited. Ideally, the final model should leverage real-time elevation data imagery provided by lunar vehicles, enabling it to instantaneously identify flat spots upon analysis.

Reinforcement Learning

2.1 Background

A more effective application of AI was a better fit for this type of project: Reinforcement Learning. Reinforcement learning is an example of machine learning, which focuses on the use of data and algorithms to imitate the way humans learn, thus gradually improving accuracy over time. Reinforcement learning is a training method within machine learning, based on rewarding desired behaviors and punishing undesired ones. The reinforcement learning model involves 2 objects: the agent, and the environment. The agent is the entity taking actions within the environment and is the learner of the system. The environment is an external system in which the agent interacts with, and receives feedback. The agent learns through these interactions through trial and error, in which we reward as positive or negative as feedback to the agent.

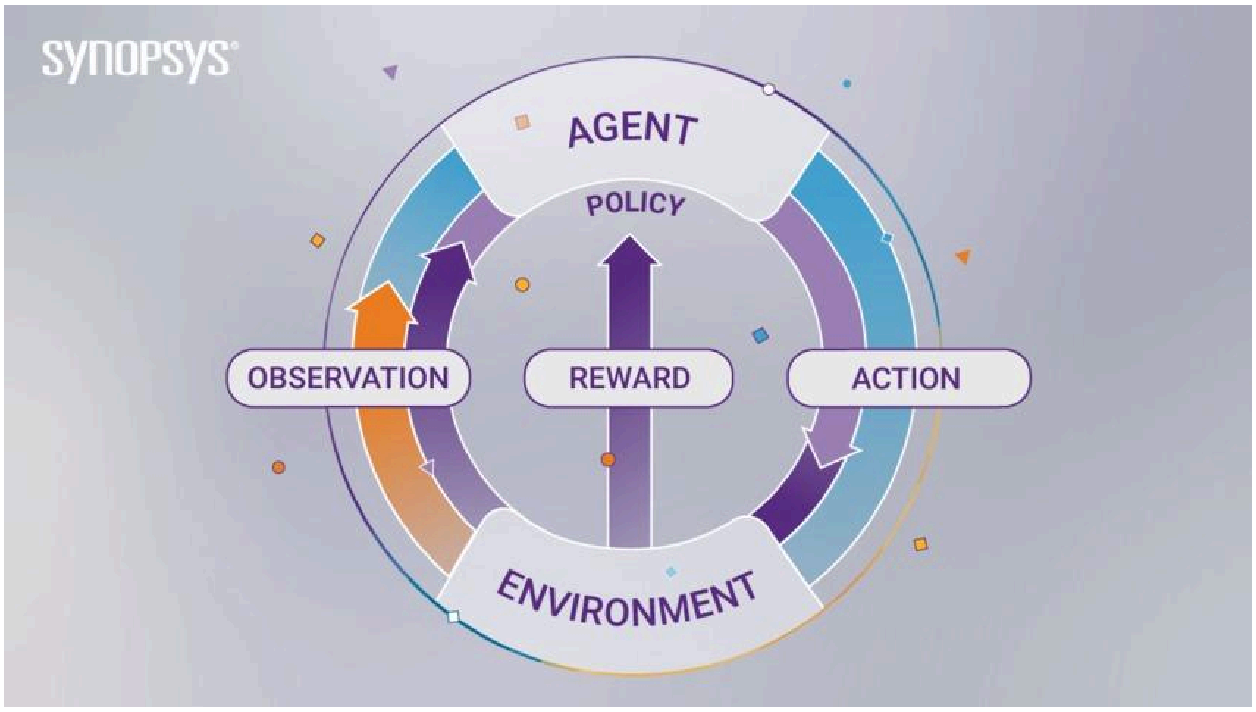


Figure 3: Reinforcement Learning Cycle [2]

The model above shows its function. The agent is placed within an environment, and from there takes observation of its surroundings. Based on the data it collects, the agent specifically chooses an action within the environment, which will be evaluated with a reward function. This ensures the agent will know if it's doing the right thing, or if it should take another action.

2.2 Custom Environment

In the MoonLander, a custom environment was built for the agent to interact through. The environment used includes a map of the moon's elevation data[3]. The photo is shaded accordingly, with each pixel attaining a numerical value of its height determined by its shade. The environment model takes the data, and converts it into an array of numbers, signifying the elevation points around an area. The observation space is the maximum area the agent is capable of exploring. Setting the dimensions of the observation space relative to the image dimensions ensures the agent will not choose a landing spot outside of what is given.

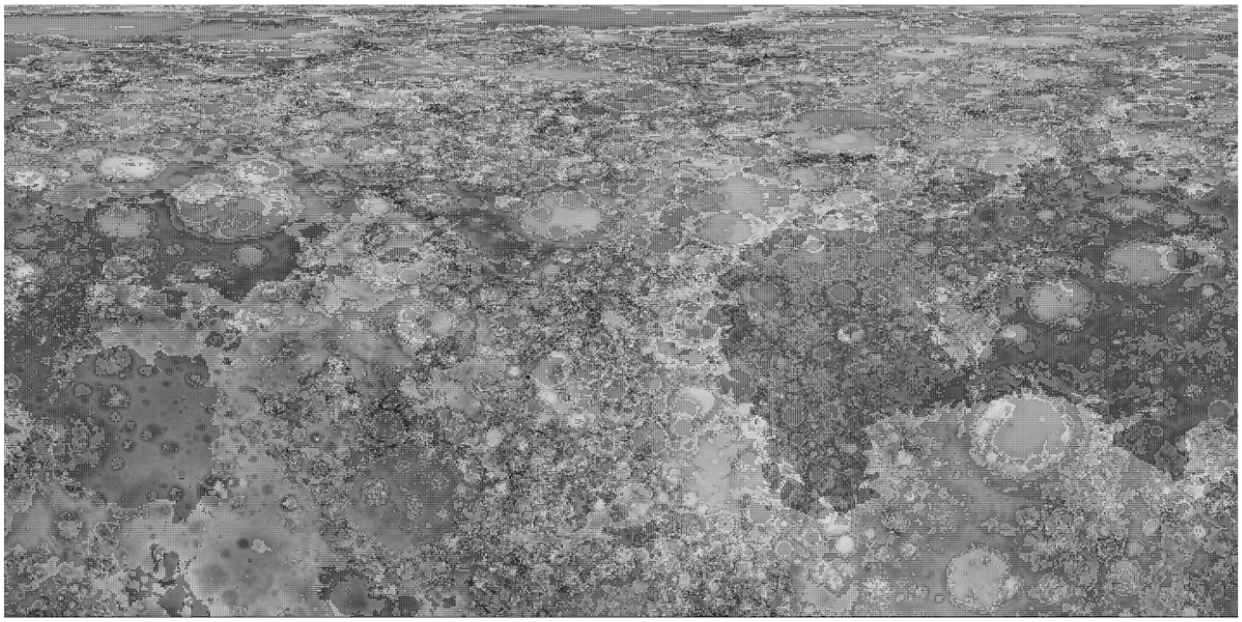


Figure 4: Elevation Image of the Moon [3]

To note, if this model was applied in a real mission scenario, a different image would be imputed into the environment. The idea would be that the space vehicle would be close enough to take an accurate picture of the moon's surface from its perspective, in which the AI model would compute a landing spot instantaneously. To simulate a real-time scenario, a random sector of the moon elevation image was zoomed into and cropped out below [5], and the AI accurately found a landing spot relative to the rover's location.

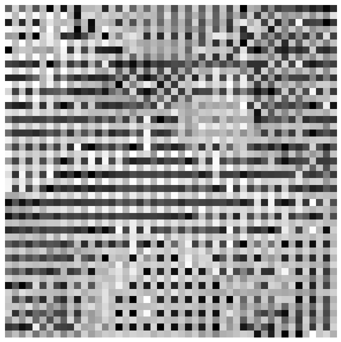


Figure 5: Closeup Example Photo of Moon[3]

2.3 Action Steps

The next section defines the agent's possible actions. A step function is defined, in which an action is passed through. The possible actions set for the agent include movement in all 4 directions (up, down, left, right). Since the environment image is already converted into an array sequence, the actions move the agent one step in the x or y direction depending on the movement, and updates its new location with an array. A 5th action is also available: Termination. This action clarifies that the agent's location is already a flat spot, and no more actions should be taken. When the agent is ready to step in, it first checks if it should be terminated, and if not, it continues taking action until the termination action has been taken.

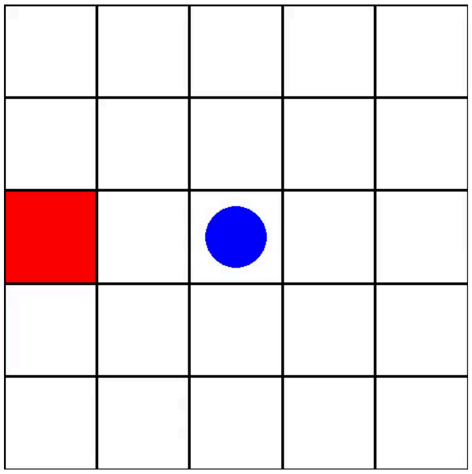


Figure 6: A GIF showing the Agent's Movement [4]

2.4 Reward Function

The reward function is the function used to calculate the cumulative reward based on the actions. It helps the agent evaluate the desirability of different states and actions. The MoonLander calculates the cumulative reward by taking into account the surrounding values of the agent's location.

def calculate_reward(self, elevation_array, agent_location, radius=1)

The elevation image array and the agent's location are called through, as well as the radius. The radius defines how far off the surrounding values it utilizes. The radius is set to 1, ensuring the reward function only scans one block in every direction of the agent's location within the array.

average_surrounding_value = (1/n)∑(surroundings_i)

Once all the surrounding values are calculated, the mean is taken to determine the average elevation for that chunk. 0

deviation = ∑ |surroundings_i - average_surrounding_va.03lue|

Using that average elevation, the deviation of the agent's value from the average is calculated. Finally, a threshold of 100 is set as a marker, and the variable reward is set to the function

(100 - deviation)/threshold

reward = max(0, (threshold - deviation) / threshold)

A 0 variable is also passed through the reward to ensure the reward stays positive, and if the agent doesn't perform well, no reward is attributed.

Training:

3.1 Setup

The environment is imported within a separate file and reset to its original state. The training model used is Stable Baselines 3, a set of improved RL algorithms based on *OpenAI Baselines[5]*. A baseline is a method that uses a variety of actions (heuristics, randomness, machine learning, etc) to create predictions for a data set. These predictions are further utilized in analyzing the model's performance. From Stable Baselines, 2 functions are imported: DQN and Evaluate_Policy. DQN is an RL algorithm that predicts the value of possible actions given a particular state. The key innovation in DQN aside from other RL algorithms is its unique use of experience replay, where the algorithm stores past experiences and random samples from the pool to update the network, thus providing a stable model. This technique was notably demonstrated in mastering video games, utilized to play with superhuman performance. In essence with DQN, an MLP Policy (Multi-Layer Perceptron Policy) is passed through, where it processes and interprets the state info from the environment to estimate the action function. Once the model is trained, the "evaluate policy" is used to evaluate the performance of the agent by calculating the mean and standard deviation for all rewards obtained during an episode. To visualize the training, the training data is plotted on a graph, signifying the iterations as X and the reward value as Y throughout training.

3.2 Training Process:

The model is trained over a given number of steps given with the function:

model.learn(total_timesteps=100000)

For the given MoonLander, it is trained over 100,000 steps each time.

Results:

Once the model has finished the training process, MatPlotLib was used to plot the reward results. As shown, the fully trained model is effectively capable of finding a flat landing spot, as indicated by the high reward. The model is also efficient in its calculation, represented by the quick reward increase in the shortest amount of steps.

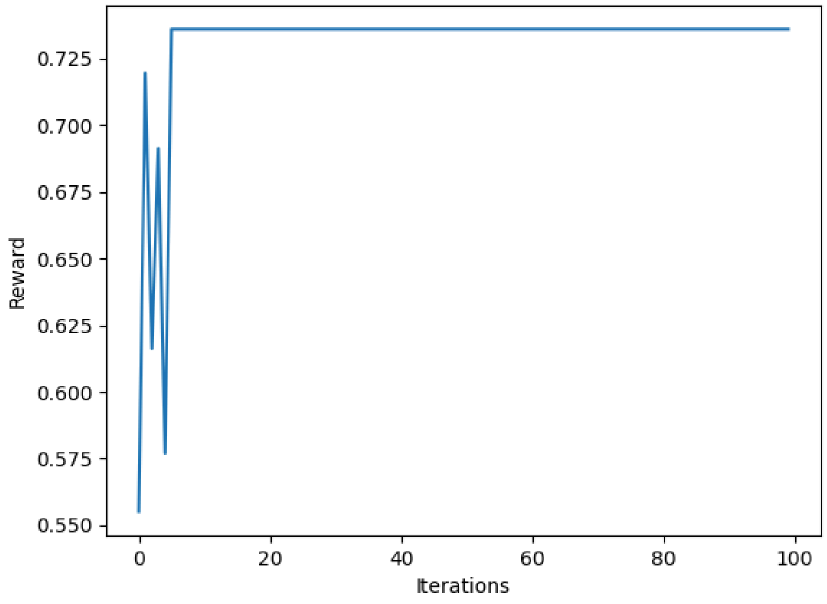


Figure 7: Training Data

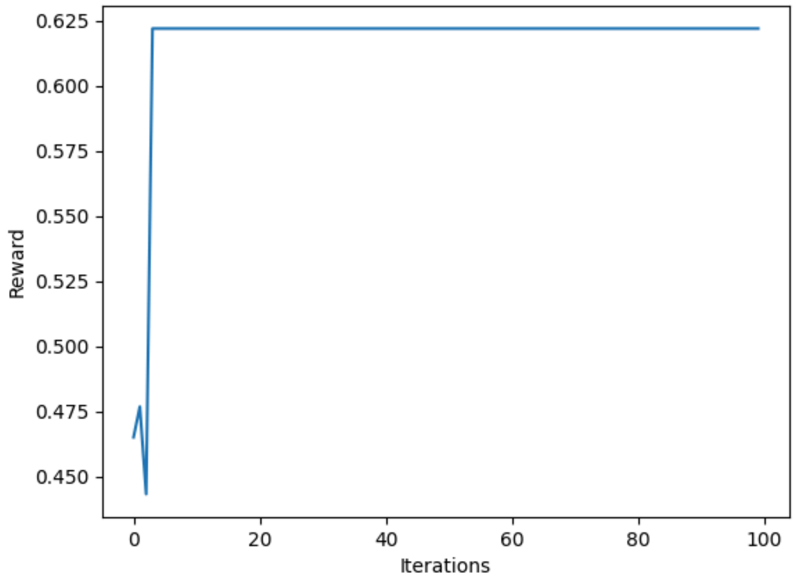


Figure 8: Training Data

Conclusion:

This study has systematically explored the integration of AI within space exploration, enhancing the precision of landing. The development and successful simulation of a reinforcement learning model represent a significant advancement in autonomous vehicle navigation. The model's capability of locating safe lunar landing zones addresses the critical challenge attempted and failed by many countries: ensuring safe and precise landing for the spacecraft.

The results of the MoonLander indicate the application of AI in space missions, particularly in challenging environments such as the Moon, can substantially increase the success rate and overall safety of these missions. The MoonLander demonstrated remarkable accuracy and efficiency in real-time scenarios, suggesting its potential application in upcoming lunar missions

The implications of this research extend beyond the lunar domain. The methodology used could be instrumental in exploring future celestial bodies, furthering our quest to unravel the mysteries of our solar system. Integrating AI in space technology not only enhances current exploration capability but opens a new horizon of learning for future planetary missions.

While the results achieved by the MoonLander were indeed promising, they represent only the initial phase in a broader journey of innovation. Looking ahead, there are significant opportunities to extend the capabilities of AI. A key area for future research involves enhancing the AI's understanding of lunar topography, moving beyond the primary objective of a safe landing. The next logical step is to train the AI with the ability to not only find optimal landing sites but also plot efficient and safe traversal paths for rovers on the lunar surface. The paths generated by the AI would be tailored to the specific goals of the real-time mission it is implied in. The utilization of AI here represents a greater step forward in lunar landing, potentially transforming how rovers are deployed and managed on extraterrestrial surfaces.

As we enter the threshold of this new era, it is crucial to recognize that the journey of exploration is ever-evolving. The potential implications of AI in space exploration are as vast as space itself, offering boundless learning opportunities. This research project serves as a beacon of light, illuminating the path toward safer and more efficient space missions. As we venture into the future, it is the synergy between human application and AI knowledge, as demonstrated by the MoonLander, that will guide us in discovering the secrets of the darkness beyond Earth itself.

Sources

[1]: Brownlee, Jason. "A Gentle Introduction to the Sigmoid Function." *Machine Learning Mastery*, https://machinelearningmastery.com/a-gentle-introduction-to-sigmoid-function/. Accessed 22 Jan. 2024

[2]: "What is Reinforcement Learning?" *Synopsys*, https://www.synopsys.com/ai/what-is-reinforcement-learning.html. Accessed 22 Jan. 2024.

[3]: "NASA's Scientific Visualization Studio." *NASA*, https://svs.gsfc.nasa.gov/cgi-bin/details.cgi?aid=4720. Accessed 22 Jan. 2024.

[4]: "Environment Creation." *Gym Library*, https://www.gymlibrary.dev/content/environment_creation/. Accessed 22 Jan. 2024.

[5]: OpenAI Baselines: DQN." *OpenAI*, https://openai.com/research/openai-baselines-dqn. Accessed 22 Jan. 2024.

[6]: Kooser, Amanda. "NASA's Artemis moon missions explained." *Mashable*, https://mashable.com/article/nasa-moon-landing-space-artemis. Accessed 22 Jan. 2024.

[7]: Berger, Eric. "Why NASA hasn't sent astronauts to the moon in more than 45 years." *Business Insider*, https://www.businessinsider.com/moon-missions-why-astronauts-have-not-returned-2018-7. Accessed 22 Jan. 2024.

[8]: Rogers, Adam. "A Crashed Israeli Lunar Lander Spilled Tardigrades on the Moon." *Wired*, https://www.wired.com/story/a-crashed-israeli-lunar-lander-spilled-tardigrades-on-the-moon/. Accessed 22 Jan. 2024.

[9]: Rouse, Margaret. "AI (Artificial Intelligence)." *TechTarget*, https://www.techtarget.com/searchenterpriseai/definition/AI-Artificial-Intelligence. Accessed 22 Jan. 2024.

[10]: "Artemis." *NASA*, https://www.nasa.gov/specials/artemis/. Accessed 22 Jan. 2024.

[11]: VChandler, David L. "Explained: Neural networks." *MIT News*, Massachusetts Institute of Technology, 14 Apr. 2017, https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414. Accessed 22 Jan. 2024.