A Neural Network-Based Proposal for Optimizing Battery Life and Autonomous Task Management in Nanosatellites

Carlos Padilla
enigmak9@protonmail.com

ICN, UNAM

April 11, 2024

LINX

Back to TOC

Table of Contents

Energy Efficiency Requirements

The network must ensure that the battery's state of charge (SoC) remains above 20%, avoiding long periods below 30% to ensure the satellite's longevity and effectiveness.

Back to TOC

Reinforcement Learning

Reinforcement learning is an area of machine learning inspired by behavioral psychology, which focuses on determining what actions a software agent should choose in a given environment to maximize some notion of cumulative reward or prize.

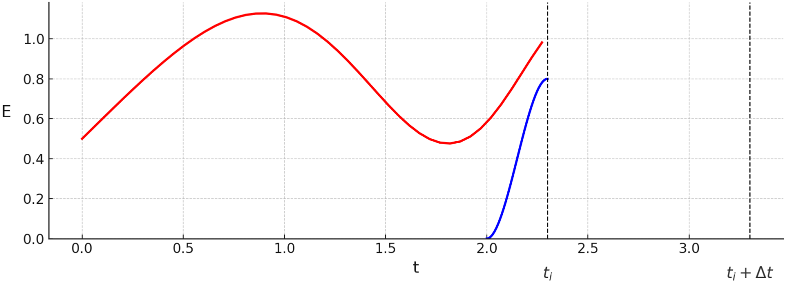
Back to TOC

DQN

Our model will be a feedforward neural network that takes the difference between the current and previous screen patches. It has two outputs, representing Q(s, left) and Q(s, right) (where s is the network input). Effectively, the network tries to predict the expected return of performing each action given the current input.
Reinforcement Learning (DQN) Tutorial

Back to TOC

W_0



Back to TOC

Project Execution Model

$$\mathrm{M}_j\left(\left[I, t'_j, \Delta t_j, P_j^R, P_j^D\right], \left[W_l^R, W_l^D\right]\right)$$

Back to TOC

Definitions PEM

Where:

- I - Time when distribution that takes over the initial time
- P_j^R - Priority of execution
- P_j^D - Priority of data download
- $\left[W_i^R, W_i^D\right]$ - Intrinsic properties

Back to TOC

Status and Vectors

Status information:

$$E(t_i); \quad \frac{dE}{dt}; \quad \frac{d^2E^-}{dt^2}$$
$$W(t_i); \quad \frac{dW^-}{dt}; \quad \frac{d^2W^-}{dt^2}$$

Data vector for project:

$$D = (d_1, d_2, \ldots, d_J) \quad \text{amount of data of task } j$$

Execution status vector:

$$S = (s_1, s_2, \ldots, s_J) \quad / \quad s_j = \begin{cases} 0 & \text{if } j \text{ not executed} \\ 1 & \text{if } j \text{ is executed} \end{cases}$$

Back to TOC

Expected vector output

$$\left[\begin{array}{c|cccccc}
 & 1 & 2 & 3 & 4 & 5 & \cdots \\
\hline
t1 & 0 & 0 & 1 & 0 & 0 & \cdots \\
t2 & 1 & 0 & 0 & 0 & 0 & \cdots \\
t3 & 0 & 1 & 0 & 0 & 0 & \cdots \\
t4 & 0 & 0 & 0 & 0 & 1 & \cdots \\
t5 & 0 & 0 & 0 & 1 & 0 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}\right]$$

Back to TOC

Bellman's equation

$$V(s) = \max_a \{R(s, a) + \gamma V(s')\} \tag{1}$$

Back to TOC

Explanation of the Bellman Equation

The Bellman equation is defined as:

$$V(s) = \max_{a} \{R(s, a) + \gamma V(s')\} \tag{2}$$

Where:

- $V(s)$ is the value of state s. It represents the maximum expected return when the agent is in state s.
- $\max_a$ denotes that we are selecting the action a that maximizes the following value.
- $R(s, a)$ is the immediate reward received for taking action a in state s.
- γ is the discount factor, balancing the importance of immediate rewards versus future rewards. It ranges from 0 to 1.
- $V(s')$ is the value of the subsequent state s', representing the expected return from that new state.

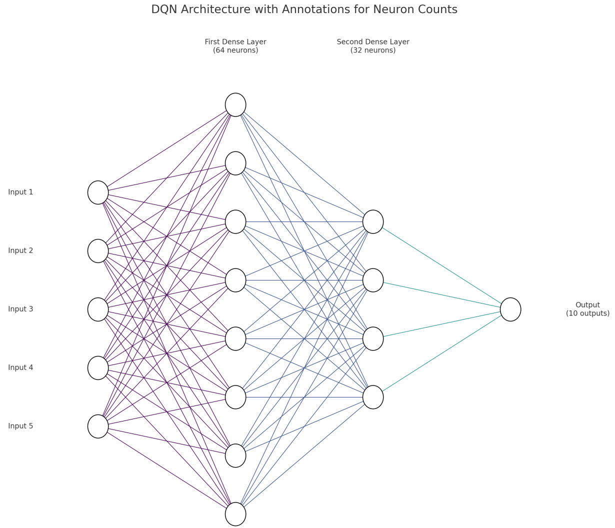
Back to TOC

Implementations

Retask priorities
Red 002
Red Plus 001
Google Colab Link
Red 003
Potencia
Nuestra interpretación para resolver el problema
SoC Cycles prediction

Back to TOC

Architecture



Back to TOC