EasyChair Preprint

№ 10783

Metaverse in InterPlanet Internet: Design of Robotic Microcontroller in Space Robots for Diverse Space Applications Using an Automated AI Learning Agent

Poondru Prithvinath Reddy

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

August 27, 2023

Metaverse in InterPlanet Internet: Design of Robotic Microcontroller in Space Robots for Diverse Space Applications Using an Automated AI Learning Agent

Poondru Prithvinath Reddy

ABSTRACT

The interplanet internet is a conceived computer network in space, consisting of a set of network nodes that can communicate with each other. These nodes are the planet’s orbiters (satellites) and landers (e.g. robots, autonomous machines, etc.) and the earth ground stations, and the data can be routed through Earth’s internal internet. As resource depletion on Earth becomes real, the idea of extracting valuable elements from asteroids or using space-based resources to build space habitats becomes more attractive, one of the key technologies for harvesting resources is robotic space mining (minerals, metals, etc.,) or robotic building of space settlement. The metaverse is essentially a simulated digital environment mimicking the real world. The metaverse would be something very similar to real world planetary activities where users (space colonies or internet users on Earth) interact with overlaying objects represented by robots, drones, etc. for real-world planetary activities like space mining, building space settlements, etc. in a completely virtual manner. Here we show how microcontroller on space robots may be designed for capturing robotic controls with different make-up for executing diverse space applications. For this, an AI agent is designed to learn to optimize the final control generation from the space operational requirement/environment. We designed an RL agent to add or to remove the controls to maintain a correct movement, actuators activation and high-performance space execution and to build through a series of steps (adding or removing controls) for improving the execution performance & efficiency of space related applications. For this we used fully convolutional neural network the Q-learning algorithm (an RL algorithm) for space applications and the algorithm trained the microcontroller design agent using a matrix representation for operational requirement. Since we have learning models of robotic shapes along with a learning agent for microcontroller design, we show an implementation of combining shape patterns and device controls with space exploration activity by means of a small model in obscene of real-world model of Metaverse for autonomous space operations. In this way, the desired response or generator loss was defined, and new environmental conditions and robotic selection patterns were synergistically combined with automated controls in learning agent for diverse space related outcomes. The results of the study simulated on existing internet here on Earth show that the real individual behaviour on a distant planet can be achieved provided the interplanet internet is available as pathway communication and undertaking of space related activities with varied robotic make-up and microcontrollers using deep learning models could be of reality even in interplanet environment.

INTRODUCTION

Inter-planetary exploration, be it Lunar habitation, asteroid mining, Mars colonization or planetary science/mapping missions of the solar system, will increase demands for inter-planetary communications. The movement of people and material throughout the solar system will create the economic necessity for an information highway to move data throughout the solar system in support of inter-planetary exploration and exploitation. The communication capabilities of this interplanet information highway need to be designed to offer; 1) continuous data, 2) reliable communications, 3) high bandwidth and 4) accommodate data, voice and video.

The interplanetary Internet is a conceived computer network in space, consisting of a set of network nodes that can communicate with each other. These nodes are the planet's orbiters (satellites) and landers (e.g., robots), and the earth ground stations. For example, the orbiters collect the scientific data from the Landers on Mars through near-Mars communication links, transmit the data to Earth through direct links from the Mars orbiters to the Earth ground stations, and finally the data can be routed through Earth's internal internet. Interplanetary communication is greatly delayed by interplanetary distances, so a new set of protocols and technology that are tolerant to large delays and errors are required. The interplanetary Internet is a store and forward network of internets that is often disconnected, has a wireless backbone fraught with error-prone links and delays ranging from tens of minutes to even hours, even when there is a connection. In the core implementation of Interplanetary Internet, satellites orbiting a planet communicate to other planet's satellites. Simultaneously, these planets revolve around the Sun with long distances, and thus many challenges face the communications. The reasons and the resultant challenges are: The interplanetary communication is greatly delayed due to the interplanet distances and the motion of the planets. The interplanetary communication also suspends due to the solar conjunction, when the sun's radiation hinders the direct communication between the planets. As such, the communication characterizes lossy links and intermittent link connectivity. The graph of participating nodes in a specific planet to a specific planet communication, keeps changing over time, due to the constant motion and the Interplanetary Internet design must address these challenges.

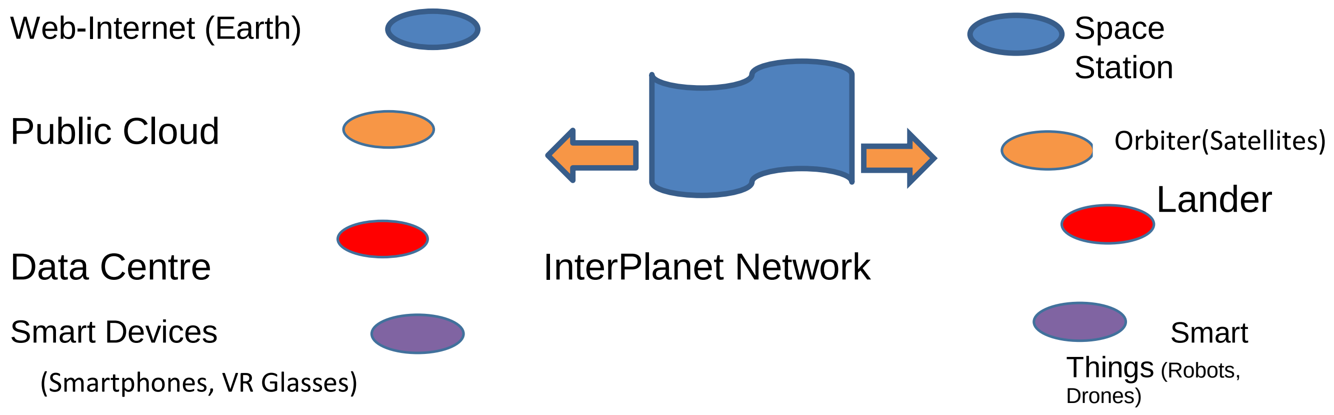
NETWORK ARCHITECTURE

A **Computer Network Architecture** is a design in which all computers in a computer network are organized. An architecture defines how the computers should get connected to get the maximum advantages of a computer network such as better response time, security, scalability, etc.

Network architecture refers to the way network devices and services are structured to serve the connectivity needs of client devices.

- Network devices typically include switches and routers.
- Types of services include DHCP and DNS.
- Client devices comprise end-user devices, servers, and smart things.

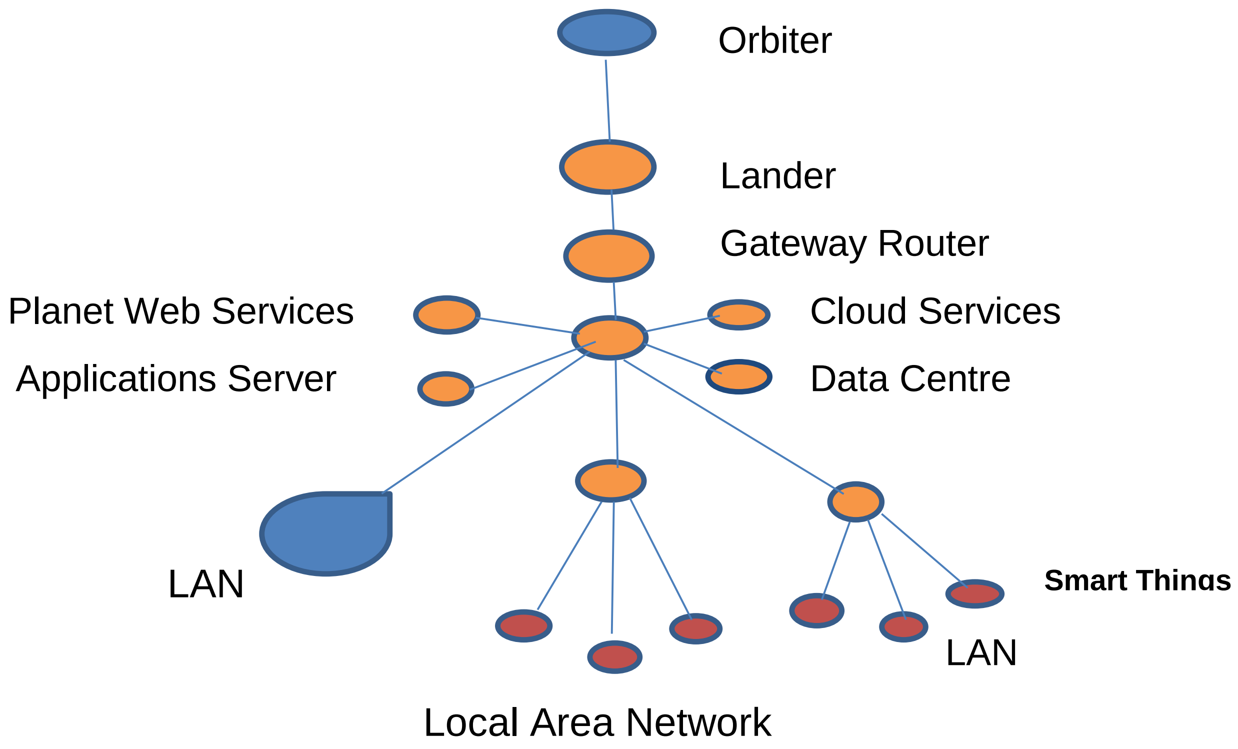
The network architecture for the planet Mars or the Moon is as shown in below figure:-



Computer networks are built to serve the needs of certain functionality and also their clients. Described below are three types of planetary networks:

- Access networks, for campuses and local areas, are built to bring machines and things onboard, such as connecting robots, drones, etc. within a location.
- Networks for data center connect servers that host data and applications and make them available to smart devices.
- Wide-area networks (WANs) connect robots and others to applications, sometimes over long distances, such as connecting robots to cloud applications related to space mining operations.

We give below the architecture of network on the planet Mars or the Earth's Moon is as shown in below figure:-



An Internet is a "network of networks" in which routers move data among a multiplicity of networks with multiple admin. domains.

The main aim of networks is to connect remote endpoints with end-to-end principle and network should provide only those services that cannot be provided effectively by endpoints.

Since the networks are predominantly wireless, the fundamental impact of distance due to speed-of-light delays and impact on interactive applications – for both data and control is to be considered. Also power consumption of wireless links as a function of distance is to be examined.

The interplanetary internet is a conceived networks of nodes and these nodes are space station, planet’s orbiters (satellites), planet’s landers, robots (drones, autonomous machines, etc.), earth ground stations and earth’s internal internet.

METHODOLOGY

Outer space contains a vast amount of resources that offer virtually unlimited wealth to the humans that can access and use them for commercial purposes. One of the key technologies for harvesting these resources is robotic mining of minerals, metals, etc. The harsh environment and vast distances create challenges that are handled best by robotic machines working in collaboration with human explorers. Humans will visit outposts and mining camps as required for exploration, and scientific research, but a continuous presence is most likely to be provided by robotic mining machines that are remotely controlled by humans either from Earth or from local space habitat.

Future **Moon(or Mars)** bases will likely be constructed using resources mined from the surface of the Moon/Mars. The difficulty of maintaining a human workforce on the Moon(or Mars) and communications lag with Earth means that mining will need to be conducted using **collaborative robots** with a high degree of autonomy. Therefore, the utility of autonomous collaborative robotics(with thousands of robots in operation) towards addressing several major challenges in autonomous mining in the lunar(Martian) environment with navigation in hazardous terrain and delicate robot interactions to achieve effective collaboration between robots and long-lasting operation.

Collaborative Robotics

Robots can be shaped to perform specific tasks. Robots have been designed and shaped in such a way that they can walk, push pellets, carry payloads, and work together in a group or possibly, to build a space settlement. They can survive for long-time without recharge and heal themselves after any damage/confusion. The shape of a robot's body and its distribution of legs and structure are automatically designed in simulation to perform a specific task, using a process of trial and error.

The methodology is essentially fundamental for getting the space robots as autonomous as possible and also as fast & optimized and the aim is to design processes involving machine learning to represent computations and their structural patterns from learning agent in realizing the desired execution in space environment. Therefore, we use robotic extraction process from the lot of thousands of robots to speed up the synthesis generation and also desired robotic shapes required in execution of diverse space related applications. The machine learning systems are required to be trained separately using reinforcement learning algorithm to arrive at the robotic designs that can easily be adopted and customized for the environment in space related applications and these are used to localize the requirement.

The methodology primarily consists of following parts:-

1. Designing neural networks on the exploration requirements of space related activities (Space mining, Building space settlements, etc.).
2. Designing machine learning systems for Extracting structural patterns from robotic space at each exploration step.
3. Introducing learning agent in the processes(space related work & robotic shapes) that uses deep neural network with learning algorithm
4. Similarly, introducing learning agent in the processor(CPU & GPU) and microcontroller that uses deep neural network with learning algorithm

5. The neural network used by the learning agents including the neural network used by the learning agent(processor and microcontroller) will be trained with learning algorithm by using different methods
6. Measuring the outcome with generator loss or optimization steps
7. Based on generation requirements, get the device control requirements of the microcontroller on the basis of process control and exploration synthesis data.
8. Similarly get the device control parameters on the microcontroller on the basis of structural pattern in the exploration step

9. Based on exploration requirements, get a series of robotic requirements that are regularly required in a particular exploration activity on the basis of computational data.

10. Similarly get the robotic-batch required out of lot on the basis of structural pattern in the exploration step
11. Carryout computational data association with the sensors and device parameter data with the graphical data of the exploration

step by matching with the desired data of completed exploration in the database.

ARCHITECTURE

1. Augmented Reality

The word ‘augmented’ means to add. Augmented reality uses different tools to make the real and existing environment better and provides an improved version of reality.

As Augmented Reality (AR) technologies improve, we are starting to see use cases and these include product visualization. There are AR apps that allow a customer to place virtual furniture in their house before buying and it is also a powerful tool for marketing as it allows users to try products before buying.

At its core, AR is driven by advanced computer vision algorithms that compares visual features between camera frames in order to map and track the environment. But we can do more. By layering machine learning systems on top of the core AR tech, the range of possible use cases can be expanded greatly.

Augmented Reality (AR) can be defined as a system that incorporates three basic features: a combination of real and virtual worlds, real-time interaction, and accurate 3D registration of virtual and real objects

2. Camera Representation

A camera is a device that converts the 3D world into a 2D image. A camera plays a very important role in capturing three-dimensional images and storing them in two-dimensional images. And the following equation can represent the camera.

x=PX

Here x denotes 2-D image point, P denotes camera matrix and X denotes 3-D world point.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

The above is vector representation of x=PX [1].

The camera representation method is frequently used in image processing and is intended to identify the geometric characteristics of the image creation process. This is a vital step to perform in many computer vision applications, especially when metric information on the scene is needed.

3. *Metaverse Algorithm*

1. **Physical Reality Modeling -** required information
   - The goal of the agent/robot
   - What the robot sees, Materials & location
   - Real Simulation for Task Execution
2. **Task Execution (Simulation)**
   - Generating actual materials (how materials arrive at the site)
   - Robots arrive in the environment (speed and goal)
   - Task Execution (Simulation Steps), is updated as the work process progresses in line with the simulation
   - Task execution performance, as we have fully functional simulator and to make a realistic system, we would like to see how well it performs and mirrors real world execution(Artificial Intelligence)
   - Implementation of Graphical Version of the Task Execution

Models for Metaverse & Algorithm

Minimum amount of required information

- The current state of the robot/agent and its environment
- The goal of the agent/robot
- What the agent sees, materials & it’s location

Agents – Attributes

We opt for the agents and they have the attributes: the sight and the goal. While the goal is chosen randomly when an agent arrives on the location, the sight is always fixed to the some value. The other noticeable fact is that our learning agents do not have a desired speed. We define the autonomous robots as entities whose primary concern is to avoid failure; they should consequently not exhibit any preference for a certain speed as long as they are working safely. Furthermore, we add an attribute to these learning agents; this is their probability of choosing a random action at each time step.

Agents as workmen

Given that we define learning agents the same way as the type of workers, we can seamlessly add them at the location. The only difference is how they will choose an action: by using their learning model, a neural network. We can therefore adapt the site's time step's algorithm to take the learning agent into account for the observation step. To decide what action it should take, the learning agent uses a neural network to approximate the Q-function. Thus, at every time step t, the agent c observes its state $s_{c,t}$; this state is then processed in some way so that it can be passed to a neural network whose outputs correspond to all the possible actions. The values of these outputs are the estimated Q-values, Q($s_{c,t}$, a); as it is using a neural network θ, we denote the Q-function approximated with that network by Q(s,t;θ). The agent then uses an ϵ-greedy strategy to choose the action $a_{c,t}$.
The neural network used by the learning agent will be trained with learning algorithm by using different methods.

Neural Network Models

Presently different neural network models are available that we will use to train our autonomous robots. These models define what information the learning agents use and how they are encoded as inputs to the neural networks. Before we start with our model, we need to define the building structure; how these neural networks are used by the learning agents. We use a feedforward neural network whose outputs correspond to the possible actions. Our models define different ways of using information about the agent's current state. Thus, they either encode different information or encode the same information differently to produce the inputs.

Required Information

We start by defining the minimum amount of information that an autonomous robot should have. Consequently, the model that we design will possess these pieces of information. They are:

- The goal of the agent/robot
- What the robot sees, Materials & location
- The current location that the agent is in
- The current speed of the agent
- Real Simulation for Task Execution

Task Execution (Simulation)

- Generating actual materials (how materials arrive at the site)
- Robots arrive in the environment (speed and goal)
- Task Execution (Simulation Steps), is updated as the work process progresses in line with the simulation
- Task execution performance and to make a realistic system, we would like to see how well it performs and mirrors real world execution (Artificial Intelligence with learning algorithm)

Robotic Design and Placement

Robots are **collections of task executors** and have no brain system of their own. But they can be programmed to work autonomously and collaborate with other robots, or eventually to do other things tackling everything from space mining to deep space exploration.

Robots can be programmed as specific executor of an assigned task for a number of situations and also using artificial intelligence to figure out the best shape for the Robots to perform in group on a more consistent basis to have better control over performance of assigned work.

Using a computational model that simulates the nature of work and everything of the Robot Capability, the process yields the robotic shape best suited to ensure the shape of the actual Robots into more efficient form suitable to a particular situation/task and accordingly enables robots to gather together in their environment forming them into groups with the same capability.

The revolution of modern computing has been largely enabled by remarkable advances in computer systems and hardware, sensor technology and robotics. However, majority of today’s robots designed are not suitable for high-end space exploration, resulting in the need to speculate about how to optimize the next generation of robots for the machine learning (ML) models with high end space applications. Further, dramatically shortening the robot design/shape requirement would allow hardware to adapt to the rapidly advancing field of ML. The ML itself could provide the means to the robot design/shape requirement , creating a more integrated relationship between space exploration and ML, and a deep-learning approach that leverages existing data.

Vast arrays of robots with different make are required for complex space applications thus, improving the selection of design patterns of these autonomous robots would be critical in improving the performance and efficiency of remote space applications and use of AI to achieve high-performance execution and robotic performance relevant to the work.

In order for the AI to design with an RL agent and the technique proved that AI can not only learn to design robotic patterns from scratch but that those structural patterns are accurate and faster than designed using any of the latest validation tools. Here an AI agent could design neural graphs and such a graph is converted into a class of robots with connection (relevant shapes) using a link generator. These generated circuits are then further optimized by a physical synthesis tool.

Robotic types for task execution are built using logic and a lot of classifications/connections, should be easy, fast to reduce any delay that can be a drag on performance and consume as little power as possible.

The robotic type design is represented as a reinforcement learning (RL) task, where we train an agent to optimize the design and delay properties of robotic batch selection and for this relations are represented using grid representation with each element in the grid mapping to a graph node, and design an environment where the RL agent can add or remove a node from the connected graph.

We propose robot placement as a reinforcement learning (RL) problem, and unlike other methods, this approach has the ability to learn from past experience and improve over time. In particular, as we train over a greater number of robotic blocks, the method becomes better at rapidly generating optimized placements for previously unseen robotic blocks, and can rapidly generate optimized placements for space robots.

A fleet of robots are divided into dozens of blocks, each of which is an individual robotic module, such as a memory subsystem, compute unit, or control logic system and these blocks can be described by a graph of class components consisting of node types and graph adjacency information. The graph of type components/robots representing the composition and structural patterns, are passed through an edge based graph neural networks to encode input state. This generates the embeddings of the placed graph and the candidate nodes.

A graph neural network generates embeddings that are concatenated with the basic work meta data to form the input to the policy and value requirement of robotic design patterns for space exploration. The policy network generates a probability distribution overall possible grid cells.

Robotic Microcontroller

A **microcontroller** is a compact integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip. Microcontrollers can be used in various industrial products and the approximate components of the hardware are core, storage, peripheral interfaces, bus, interrupt module, clock module, etc.

A robot microcontroller is basically the brain of the robot. It is used to collect the information from various input devices such as sensors, switches and others. Then it executes a program and in accordance with it controls the output devices such as motors, lights and others.

Machine Learning on Microcontrollers

Using today's advanced AI systems to run machine learning on smaller devices with processors like microcontrollers offers benefits – as enablers of AI.

Microcontrollers preceded the development of CPUs and GPUs and are embedded in virtually every kind of modern device with sensors and actuators. They are a vital consideration for enterprises interested in weaving AI into physical devices, whether to improve the user experience or enable autonomous capabilities in devices.

One exciting avenue in the world of AI research and development is finding ways to shrink AI algorithms to run on smaller devices closer to sensors, motors and people. Developing embedded AI applications that run machine learning on microcontrollers comes with different constraints around power, performance, connectivity and tools. The rise of new tools like TinyML deployed on microcontrollers enables intelligence to be distributed into more connected products, whether they be smart home gadgets, toys, industrial sensors or otherwise.

The biggest difference between CPUs and microcontrollers is that microcontrollers are often directly connected to sensors and actuators. This reduces latency, which is essential in safety-critical applications like controlling brakes and industrial equipment. The big trend in the AI industry is moving machine learning inference to the edge, where the sensor data is generated i.e. making machine learning small enough to fit on edge devices. Further, the microcontrollers act as low-end CPUs with limited processing capability and they have two crucial advantages: low cost and low power consumption.

One way of deploying AI on a low-power microcontroller – is a new way of creating microcontrollers with integrated neural processing units (NPUs), which are specialized units designed to run machine learning models on microcontrollers efficiently. These generally come with specialized SDKs that can transform neural networks prepared on a computer to fit onto an NPU. These tools generally support models created with frameworks like PyTorch, TensorFlow and others.

Here we used small system configurations that can transform neural networks prepared on a computer and to fit efficiently onto a low-power microcontroller with integrated neural processing capability

In order to fit an NPU, engineers often need to prune a model or adjust its architecture for the NPU, which requires a lot of expertise and extends the development time. Further, developers also need to weigh the tradeoffs between the lower cost of microcontrollers compared with CPUs or GPUs and their flexibility and also It's harder to reconfigure or retrain embedded systems quickly.

Therefore, a pruned model solution using microprocessors will sometimes make more sense, with a use case of microcontrollers at the edge enabled with processor to combine information from a variety of sensors to determine when a complex piece of equipment such as Space Robot needs operating level and also to determine if some combination of conditions is a likely fit to space exploration and also to monitor as well as activate devices based on trigger.

Alternatively, as the control requirements are limited in scope, we can even design a mathematical model for robotic controls to fit into low power microcontroller to enable them to work in a limited or restricted environment with reduced complexity of the designed models with portable processing units.

Microcontroller Design

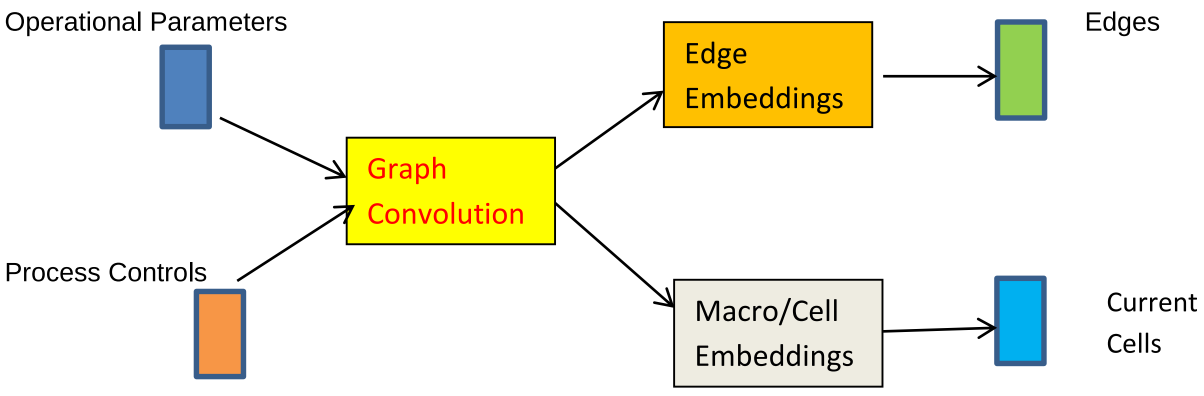
Here we look at the design aspects of microcontroller. This is nothing but designing a learning agent for improving the space exploration performance along with the desired robotic structural patterns for autonomous space execution in the form of Control Design.

In order for the AI to design with a run at RL agent and the technique proved that AI can not only learn to design controls from scratch but that those controls are faster than controls designed using the latest validation tools. Here an AI agent could design neural graphs and such graph is converted into a controls with operating parameters using a control generator. These generated controls are then further optimized by a physical synthesis tool using synthesis optimizations such as sensor sizing, actuator calibration, etc.

The device control design is represented as a reinforcement learning (RL) task, where we train an agent to optimize the operations and delay properties of controls and for this device controls are represented using grid representation with each element in the grid mapping to a graph node, and design an environment where the RL agent can add or remove a node from the control graph.

We propose device control placement as a reinforcement learning (RL) problem, where we train an agent (i.e, an RL policy) to optimize the quality of operating parameters. Unlike other methods, this approach has the ability to learn from past experience and improve over time. In particular, as we train over a greater number of control blocks, the method becomes better at rapidly generating optimized controls.

The control logic system for the microcontroller is divided into two blocks, each of which is an individual module and these blocks can be described by a graph of control components consisting of node types and graph adjacency information. The graph of process control components requisite for the exploration(operational parameters) with desired structural patterns, are passed through an edge based graph neural networks to encode input state. This generates the embeddings of the placed graph and the candidate nodes.



A graph neural network generates embeddings that are concatenated with the basic meta data to form the input to the policy and requirement of control design for space execution. The policy network generates a probability distribution overall possible grid cells onto which the current node/cells could be placed.

RESULTS

In obscene of graph databases using graph representation for machine learning systems for managing robotic generation data, we build and store the graphs in a simple read format i.e. matrix representations (stored as a node or record with edge list) to perform link prediction.

We have represented this model as matrix with encoded values with possible values for each of the nodes along with the link attributes. We populated the matrix data with randomly generated data and simulated to represent the real world robotic control elements/nodes as a link. Here we used small system configurations that can effectively transform neural networks prepared on a computer to fit efficiently onto a low-power microcontroller with integrated neural processing capability.

Here, a simple neural network model to work on low power microcontroller is designed and the system with different configurations for the hidden structures of the networks:

- 2 hidden layers: the first with 16 neurons and a *tanh* activation function; the second with 8 neurons and a *linear* activation function. No dropout.
- 2 hidden layers: the first with 16 neurons and a *tanh* activation function; the second with 8 neurons and a *linear* activation function. Dropout rate of 0.5.

The dropout rate of 0.5 has been chosen because it seems to be optimal for a wide range of networks.
The results for our CNN based model – RL policy model – The networks that do not use dropout seem to learn well. The percentage of desired generation for the networks (without dropout) is high.
Although we only have partial results, we can make the following observations: the networks that do not use dropout seem to learn well, while the network using dropout does not; it either learns very slowly or just converges to very low level of generation requirements.

CONCLUSION

The interplanetary computer network in space is a set of computer nodes that can communicate with each other. We proposed a network architecture with planet’s orbiters, landers (robots, etc.), as well as the earth ground stations and linked through Earth’s internal internet, and consisted of complex information routing through relay satellites. As we know, the metaverse will be very different from the internet of today due to massive parallelism, three-dimensional (3D) virtual space and multiple real-world spaces like space mining, building space habitats, etc. We presented a robotic shape synthesis equipped with AI-driven learning that can effectively explore unknown and complex phenomenon of applications in space environment and is also designed a microcontroller unit with RL agent (Q-learning) to add or to remove the controls to maintain a correct movement and actuation, and to build through a series of steps(adding or removing controls) for improving the performance & efficiency of space applications in an open-ended way. In this way, an automation assisted shape synthesizer with reconfigurable system that is part of Metaverse is feasible for automated execution of diverse space related outcomes depending on the applications and in that respect an implementation of Reinforcement Learning agent as a part of microcontroller unit based on small model is presented. Although the platform model with learning agents given us a method of optimizing space applications however, this need to be tested using natural allocation for real space applications.

REFERENCE

1. **Poondru Prithvinath Reddy**: **"Metaverse in InterPlanet Internet: Modeling, Validation, and Experimental Implementation", Google Scholar**