Reinforcement Learning Applied to Cognitive Space Communications

Rigoberto Roche and Janette C. Briones
NASA Glenn Research Center

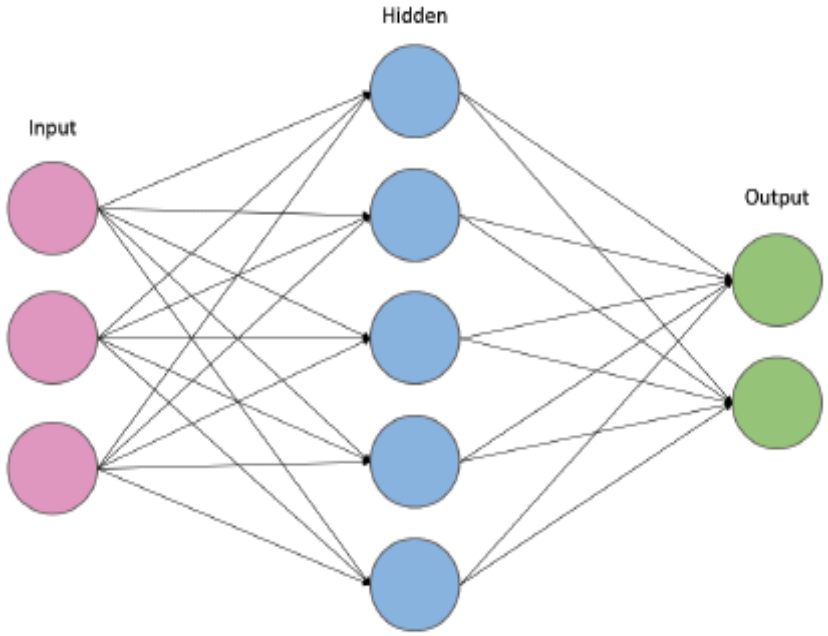
IEEE CCAA Workshop

Overview

- The future of space exploration depends on robust, reliable communication systems. As the number of such communication systems increase, automation is fast becoming a requirement to achieve this goal.
- A reinforcement learning solution can be employed as a possible automation method for such systems. The goal of this study is to build a reinforcement learning algorithm which optimizes data throughput of a single actor.
- A training environment was created to simulate a link within the NASA Space Communication and Navigation (SCaN) infrastructure, using state of the art simulation tools developed by the SCaN Center for Engineering,
- Reinforcement learning was then used to train an agent inside this environment to maximize data throughput. The simulation environment contained a single actor in low earth orbit capable of communicating with twenty-five ground stations that compose the Near-Earth Network (NEN).
- Initial experiments showed promising training results, so additional complexity was added by augmenting simulation data with link fading profiles obtained from real communication events with the International Space Station.
- A grid search was performed to find the optimal hyperparameters and model architecture for the agent. Using the results of the grid search, an agent was trained on the augmented training data.
- Testing shows that the agent performs well inside the training environment and can be used as a foundation for future studies with added complexity and eventually tested in the real space environment.

Neural Networks and Reinforcement Learning

- Neural networks are modeled after the human brain. Individual neurons are connected together and activate in response to an input to produce an output.
- Each neuron sums weighted inputs feeding into it, then passes the weighted sum through a non-linear activation function to produce its output. In this way, some neurons are activated while others are not depending on the given input.
- A reinforcement learning agent interacts with the environment over time. At each timestep, the environment provides a state to the agent which takes an action, resulting in a numerical reward and new environment state



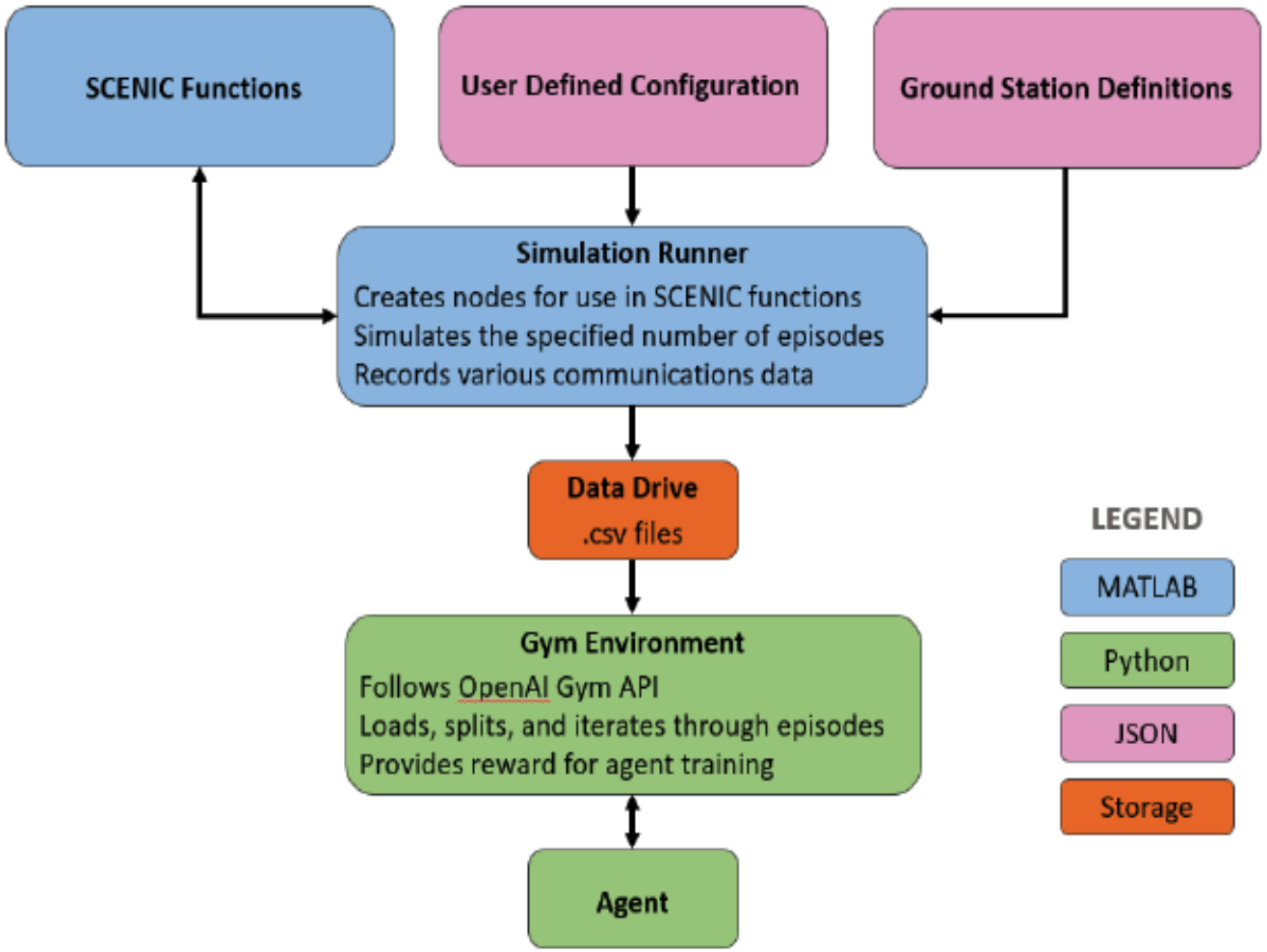
	Blocked	*Not Blocked*
Move Forward	0	1
Turn Randomly	1	0

Simulation Environment

- A custom environment was used to generate simulated links. This environment was a modification of open source Gym.
- The Gym environment continuously generates packets of a specified size in bits for the agent to downlink.
- The agent chooses a link at each time step, and the remaining bits in the packet are decremented according to the data rate of the chosen link.
- Three different numerical rewards are given to the agent: 0.1 for selecting a non-real link, 1 for selecting any real link, and 60 when a full packet is downlinked.

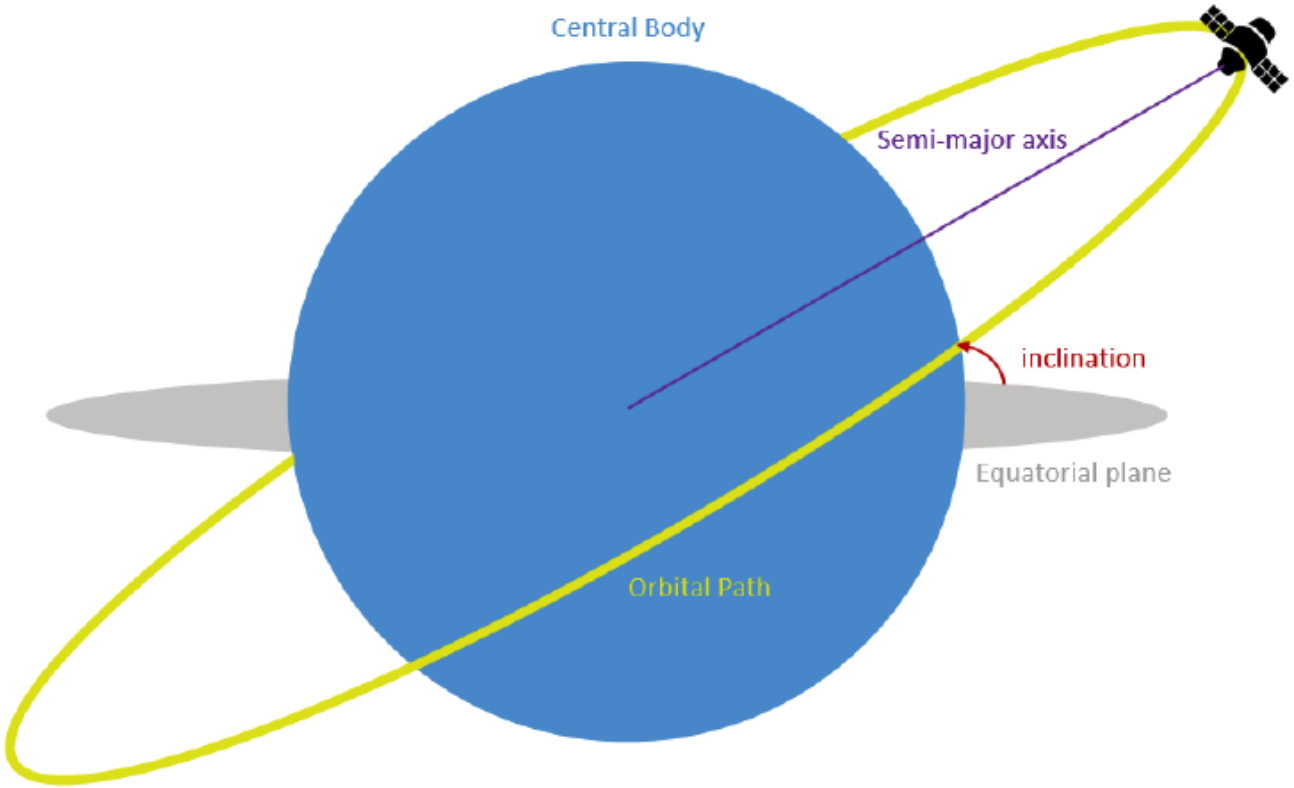
Parameter	*Value*
satX	-3294.98
satY	-6514.07
satZ	1.95
Ln_name	SGI_receive_X
Ln_x	-1524.66
Ln_y	6191.31
Ln_z	154.37
Ln_recvPower	-158.29
Ln_rangeRate	-5.83
Ln_recvEsNO	16.3
Ln_dataRate	1.69
Ln_maxBandwidth	40
Ln_BER	3.23e-11
Ln_modcod	QPSK_Uncoded

Simulation SetUP

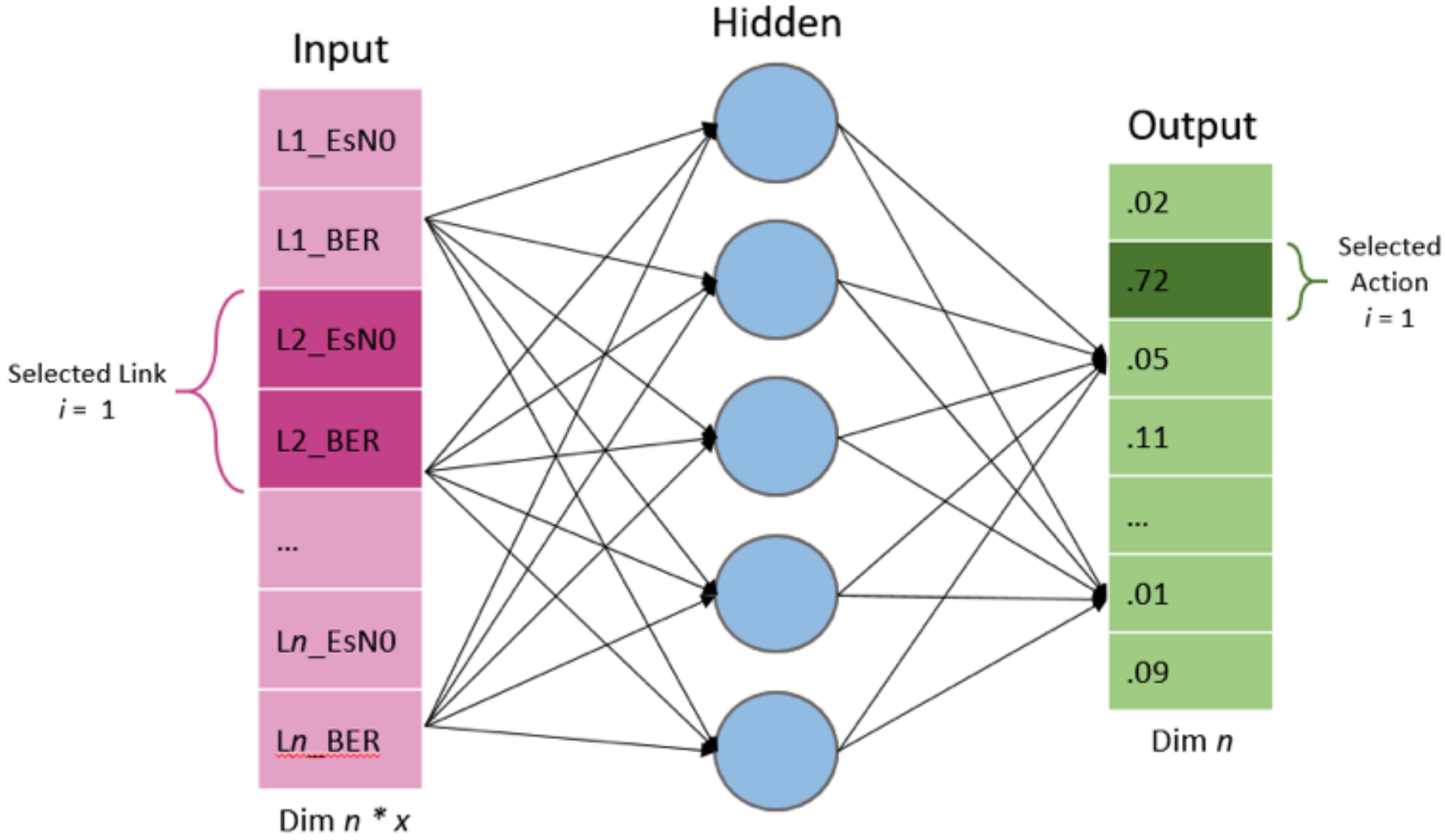


Simulation Environment (cont)

- A representative set of training episodes are necessary to effectively train an agent capable of succeeding in the real world environment.
- This was achieved by varying the starting epoch of the simulation along with the inclination and semi-major axis of the agent's orbit.
- Over 3000 episodes in total were generated using this method.



Building and Training the Agent



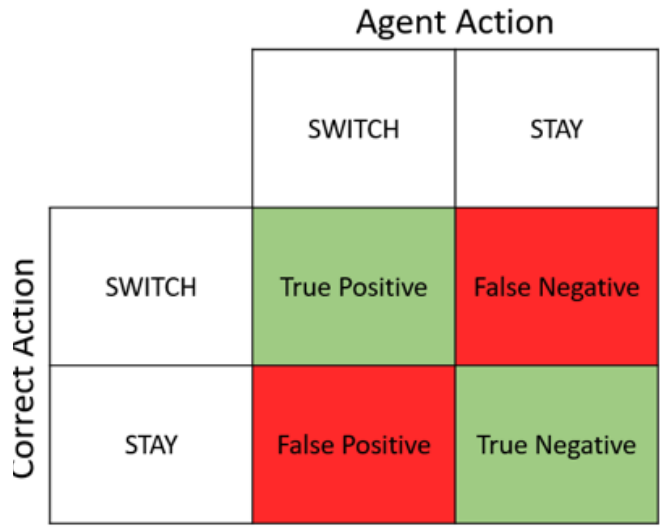
Feature Choice and Transformation

Hyperparameter	*Description*
Shape	Array defining number of hidden layers and their shapes. Ex: [64, 128, 64]
Activation	Activation function to use throughout the network. Ex: RELU
Dropout Rate	Dropout rate to use on each layer of the network. Ex: 0.5
Bias	Whether to include bias in the network or not. Ex: True
Optimizer	Gradient descent optimizer to use during training/associated learning rate. Ex: ADAM, 0.1
Batch	Size Number of timesteps to include in a single update batch. Ex: 8

Hyperparameter Tuning

- Models were compared against one another using a sum of their average reward score on the validation episodes and their recall score.
- Recall was prioritized over precision in this study because the goal of the study was to validate an agent's ability to correctly switch when it is necessary.
- Once this ability is proven, techniques can be used to mitigate the frequency of extraneous switches made by the agent.

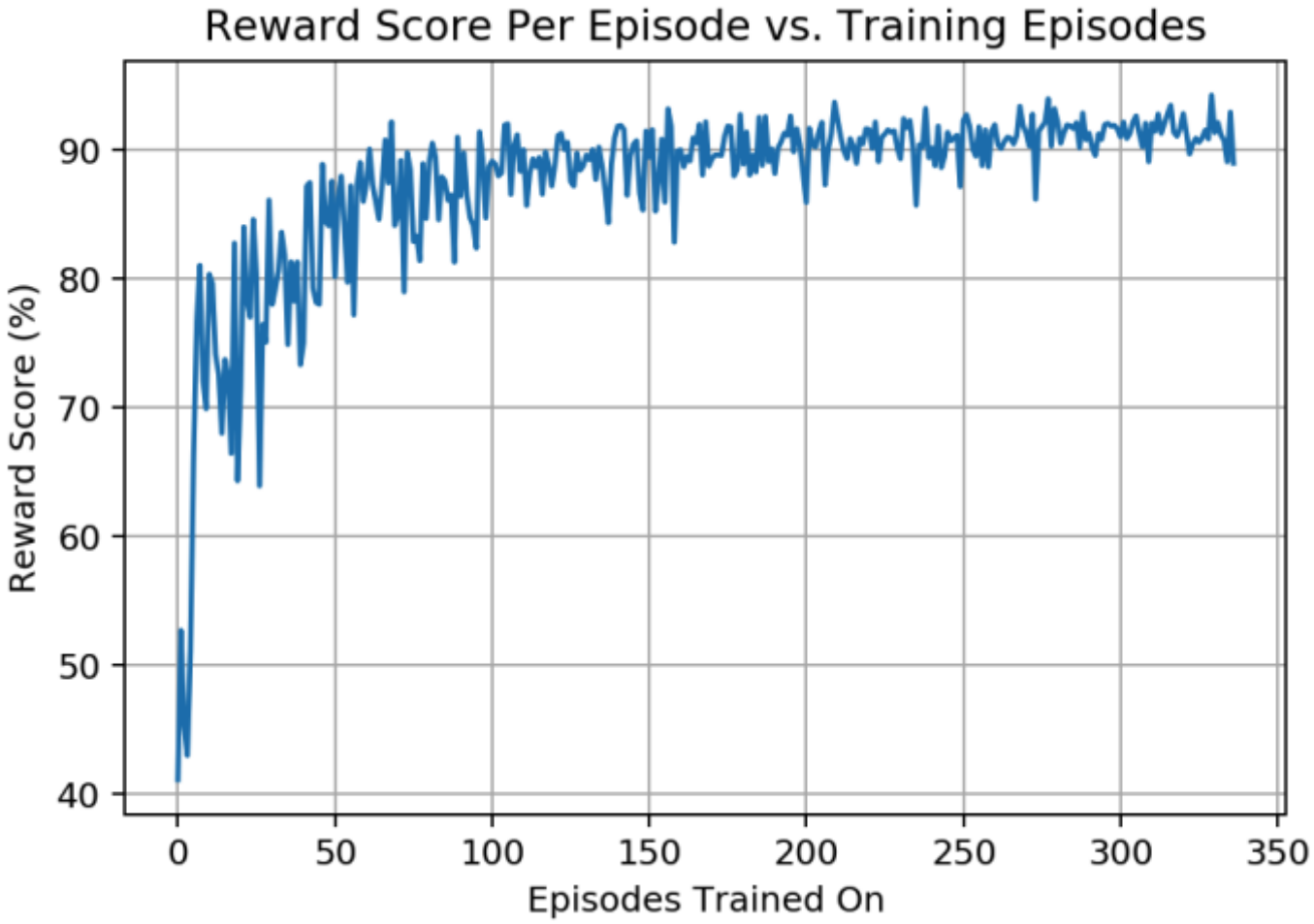
Shape	*Dropout Rate*	*Batch Size*
[64]	0.00	2
[128]	0.25	4
[64 128 64]	0.50	8
	0.75	



- The hyperparameters which produced the best model during the Grid Search were used to train a final agent on the same 450 episodes used in the Grid Search with a 75/25 train test split.
- The agent was thus trained on 337 episodes and validated on 113 episodes.

Results

Correct Action \ Agent Action	SWITCH	STAY
SWITCH	*True Positive* 78,500	*False Negative* 26,377
STAY	*False Positive* 78,787	*True Negative* 137,476



Results (cont)

- The lowest score on any episode in the validation set was 94.03%, and the highest was 100%.
- The average score was 98.12% with a recall of 74.85%, a precision of 49.91%, and F-beta score with beta = 3, of 0.713.
- A recall of 74.85% is a positive result and shows that the agent switches the majority of the time that a switch is necessary.
- The precision is very poor at under 50%. Due to the method used to augment the episodes, certain timesteps in the episodes had links which had the exact same BER. Among the 78,787 classified False Positives, 1,812 of these came from a decision by the agent to switch from one link to another with the exact same BER.
- Intuitively it is understood that these are not truly false positives but rather an issue with the definition of false positive used in this experiment

Discussions

- The environment created for this study can be used for testing other reinforcement learning agents. It can also be updated to use a different underlying simulation tool, without requiring a rewrite of agent training code, thanks to the decoupled nature of the simulation itself and agent training environment.
- There will always be drawbacks to simulating the training environment for an agent one hopes to deploy in the real world. Antenna slewing and competition for ground station resources were both excluded from the environment for simplicity in this initial research.
- These are important considerations for any cognitive link algorithm. The agent was not tested in an environment which incorporates these and thus its performance in such an environment cannot be estimated. In addition, downlink of data was simulated numerically and not using actual channels capable of introducing additional noise and interference.
- Despite these drawbacks, the environment presented sufficient complexity to the agent to ensure that results are valid indicators of the viability of reinforcement learning as a solution to the problem of cognitive links.
- The reinforcement learning techniques used in this study are simple, especially when compared with newer techniques such as Deep Q-Learning, Asynchronous Actor-Critic Agents, and Inverse Reinforcement Learning. The agent's ability to learn in the simulated environment developed for this study validates that the core methods used in reinforcement learning are applicable to the problem

Conclusions

- This study presented an alternative approach to standard methods for the task of link switching of a space orbiting asset between ground stations via a reinforcement learning approach.
- This approach was selected because it allows for scalability and generalization to other decisions within a well-defined infrastructure, such as that of a communications system.
- In this study, a reinforcement learning algorithm was built, which optimizes the data throughput of a single space asset. A training environment was created using available simulation tools developed by the SCENIC lab at Glenn Research Center, that model the closest possible representation of the real operating environment.
- The reinforcement learning algorithm was used to train an agent inside this environment to maximize data throughput.
- Results from the conducted experiments showed promising characteristics of this approach in terms of correct decision making, even in the presence of additional complexity, such as strong multipath fading in the communication link.
- The testing showed that the agent performs well inside the training environment and can be used as a foundation for future studies with added complexity and eventually it can be tested in the real space environment.