Scalable Multi-Agent Sensor Tasking Using Deep Reinforcement Learning

Peng Mun Siew[*], **Tory Smith**[†], **Ravi Ponmalai**[‡], **Richard Linares**[§]
Massachusetts Institute of Technology, The Aerospace Corporation

ABSTRACT

Satellite launches have seen a dramatic increase in recent years, driven by the growth of commercial and government constellations for a range of applications, including communication, navigation, and Earth observation. While this trend provides numerous benefits, it also puts pressure on existing narrow Field of View (FOV) ground-based sensor networks to keep pace with the growing volume of objects. Also known as the *curse of dimensionality*, wherein the complexity of the object-tracking problem grows exponentially as the number of targets and length of the observation window grows linearly, this complexity grows even further when incorporating multiple agents acting within the same environment. Current methods for allocating sensors to specific tasks are often manual, time-consuming, and prone to human error. As a result, there is a need for more robust and sophisticated methods for monitoring and managing the space environment to ensure the safe and efficient use of this valuable resource. In this paper, we propose a Multi-agent Reinforcement Learning (MARL) approach to model the relationships between Space Objects (SOs) and sensors toward a method that improves performance over previous techniques, as well as affords scalability and adaptation to novel scenarios.

1. INTRODUCTION

One of the primary tasks to achieve Space Situational Awareness (SSA) is the creation and continuous maintenance of a comprehensive catalog of Earth-orbiting SOs. This task consists of detecting, identifying, and tracking the various SOs, comprised of operational satellites, spent rocket bodies, and space debris, in order to establish their orbital tracks. The tracked SO catalog is then used to predict potential collision events and provide timely information to satellite operators when making collision avoidance maneuvers. Furthermore, the tracked SO catalog also aids in predicting reentry events and managing potential risks caused by the reentry of derelicts, spent rocket bodies, and space debris.

The creation and maintenance of the tracked SO catalog is accomplished by using a network of globally distributed ground-based radar, and optical telescopes, and a few space-based assets to systematically collect data on the positions and trajectories of SOs in Earth's orbit. This process has become increasingly challenging with the exponential surge in the number of SOs over the last decade. The increase in SOs has significantly outpaced the growth in the number of sensors, necessitating an efficient allocation of available resources to safeguard our space environment. The distinct sensor capabilities and limitations of each sensor in the SSA network as well as the sensor coverage gaps further necessitate the need for efficient sensor tasking algorithms.

The SSA sensor tasking challenge can be framed as a classical resource allocation problem, where the objective of the problem is to maximize the return within the confines of limited resources. The SSA sensor tasking problem suffers from the *curse of dimensionality*, where the complexity of the problem increases exponentially as the dimension of the problem grows. In addition, the nonlinearity in orbital dynamics, uncertain space environment, and relationships between tasking objectives, sensor constraints, and available resources further compounds the complexity of the problem [1].

In a real world context, there are often multiple sensors acting within the same environment. This inclusion of other sensors, often with overlapping Fields of Regard (FOR) adds to the challenge of the sensor tasking problem as coordination is now needed between the different agents. If the agents are not cooperating properly, it can result in redundant

[*]Research Scientist, Department of Aeronautics and Astronautics. E-mail: siewpm@mit.edu
[†]Captain, United States Space Force; Masters Student, Department of Aeronautics and Astronautics, E-mail: tsmith2@mit.edu
[‡]Senior Member of Technical Staff, Data Science and Artificial Intelligence Department, E-mail: ravi.b.ponmalai@aero.org
[§]Associate Professor, Department of Aeronautics and Astronautics. E-mail: linaresr@mit.edu

data collection, where multiple sensors observe the same information, wasting computational resources and bandwidth. Current myopic policies are ineffective in this coordination and can often lead to the aforementioned redundant collections.

Artificial Intelligence (AI) and more specifically MARL has shown its ability to compete with and surpass human and traditional coordination methods within a variety of domains from video games to collision avoidance [2, 3]. This performance has also shown to be scalable even when the MARL algorithms are trained using relatively few agents [4, 5]. Existing research has highlighted the potential of single-agent Deep Reinforcement Learning (DRL) in training narrow FOV sensors on ground and space-based platforms to effectively observe space objects in the near-Earth space environment [6–8]. This includes objects in low Earth orbit (LEO), geosynchronous (GEO) and cis-lunar orbital regimes. In these orbital regimes, sensors modeled using single-agent DRL models have achieved higher cumulative space object observations while also exhibiting lower mean covariances across all objects in the environment compared to those informed by myopic policies. This performance was also reflected when exploring scheduler-based MARL methods in relatively simple environments [8]. However, while these have shown improvement over myopic polices, these methods still suffer from a lack cooperation between agents in more complex scenarios where there is an overlap between FOR and information is shared between neighboring agents.

In this paper, we explore a decentralized MARL algorithm for efficiently coordinating a group of sensors to meet the complexities of the multi-agent sensor tasking problem. We demonstrate a scalable solution such that agents trained in an environment with relatively few SOs and in concert with a small number other agents are be able to perform in more complex environments with a larger number of SOs and other agents.

The paper is organized as follows: The SSA environment formulation is presented in Section 2. In Section 3, the MARL formulation and the MARL agent architectures studied in this paper are presented. The training statistics, performance, and robustness analysis of the trained agents are presented in Section 4. Finally, the paper is concluded and suggestions for future work are provided in Section 5.

2. SSA ENVIRONMENT FORMULATION

A custom asynchronous multi-agent space situational awareness (SSA) environment is developed using the OpenAI Gym library [9] and based on the previous works in [1, 8]. Our SSA environment efficiently tracks the states of the SOs, generates noisy measurements within the FOV of each sensor, updates the SOs' covariances, and generates the instantaneous reward of a taken action. A 3D representation of the environment can be seen in figure 1, where a network of three ground-based sensors are cooperating to maintain the SO catalog. The various SOs are represented by color-coded cubes; blue-active satellite, red-space debris, and green-spent rocket bodies. The hemispheres corresponds to the FORs of the respective sensors at different maximum viewing ranges showing that even with a small number of globally distributed ground-based agents, there is a large area of overlap between the sensors' FORs.

Each environment step occurs when one of the agents has completed its action. The agent will then utilize the current environment observation to select its next action. The environment observation refers to the state of the environment that is accessible to the agent during the decision-making process. When an agent chooses an action, the amount of slew time it will need to complete its action is calculated and placed into a queue, which manages the order of agent actions. The next agent to receive an observation and reward is the one whose action is projected to be completed in the shortest time span since the last action was completed. This ensures that the agent actions are evaluated in the correct sequence. The environment is then propagated forward in time to the next nearest action completion time. SOs that fall within the acting agent's FOV are then identified. For each of these SOs, a measurement is generated based on the selected measurement (sensor) model. The states and covariance of the observed SOs are subsequently updated using these new measurements. Following the measurement update, a new environment observation and instantaneous reward is formulated by the environment. The acting agent's action, observation, and reward are then passed to the model for training and the process is repeated until a termination criterion is reached.

Each episode rollout is limited to a finite horizon scenario, typically spanning a specific duration. SO state propagation is done via the Simplified General Perturbations 4 (SGP4) model using the open source PyEphem library [11]. The SGP4 model was first introduced in 1969 and remains one of the most prevalent used model for propagating SOs in the Earth's orbit [12]. An Unscented Kalman Filter (UKF) [13] formulation is employed to propagate and update the covariances of the SOs. In our study, the episode rollout is limited to 5400 seconds (1.5 hours).

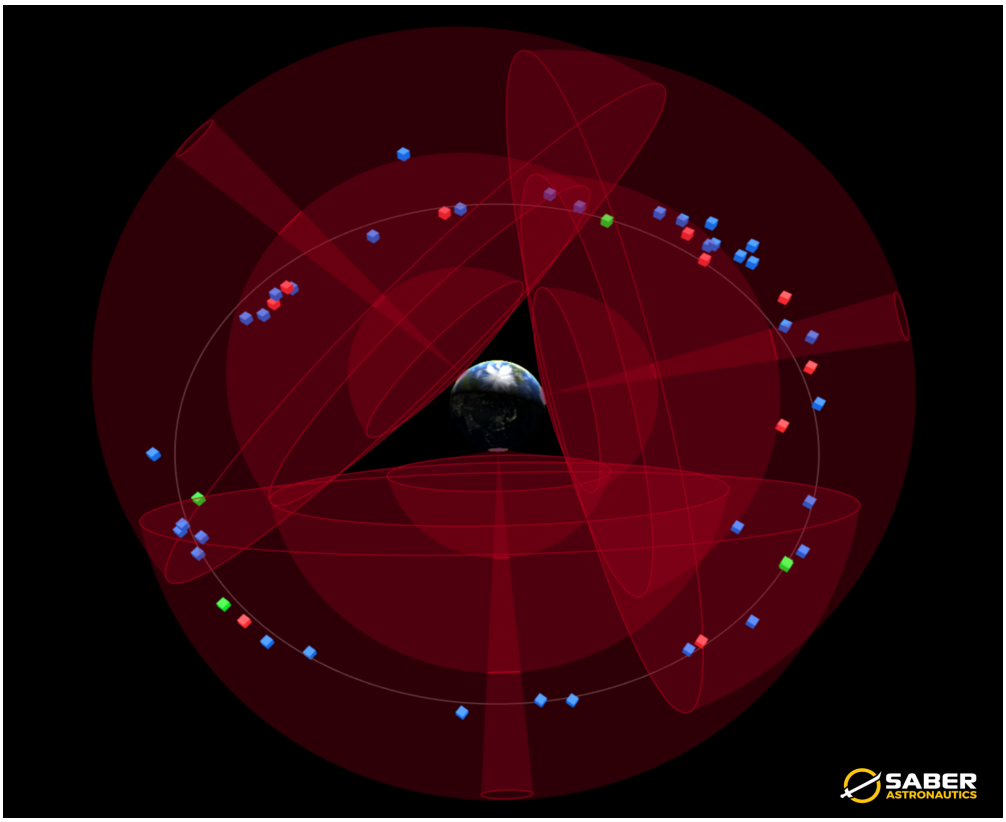


Fig. 1: Environment Representation in Space Cockpit [10]

2.1 Sensor Parameters

Each sensor is modeled as a simplified ground-based optical sensor with an integrated Light Detection And Ranging (LiDAR) system, allowing the sensor to provide Cartesian coordinate measurements for all SOs within its FOV. The sensor is assumed to be able to accurately assign each measurement to the corresponding SO. The specifications of the optical sensor are modeled with reference to the Zimmerwald SMall Aperture Robotic Telescope (ZimSMART), an operational telescope managed by the Astronomical Institute of the University of Bern, Switzerland [14]. The sensor parameters are outlined in Table 1. When taking an action, the initial 4° of slew angle is approximated to take 9s, which includes both the time for the system to stabilize and the subsequent dwell time, while subsequent 4°s of slew requiring 4.55s. The mount of the sensor utilizes independent motors for azimuth and elevation control. Thus, the action slew time can vary between 9s to 209.2s depending on the required slew angle.

Table 1: Sensor Parameters

Sensor Parameter	Values
Field of View	4°
Minimum Viewing Horizon	14°
Actions Slew Rate	9s/first 4°
	4.55s/subsequent 4°

Conventional ground-based sensors can point to any arbitrary direction above their respective minimum viewing horizon, resulting in infinitely many pointing directions within their FOR. In the context of our work, this continuous action space is discretized into non-overlapping patches that span the sensor's azimuth and altitude range. This discretization of the FOR allows us to reduce the complexity of the action space, by transforming the continuous action space into a finite-set of actions.

2.2 Sensor Locations

Sensor locations were based on publicly available current or past sensor locations of the United States Space Force (USSF) Space Surveillance Network (SSN) [15], [16], [17], [18], [19], [20], [21]. These are listed in table 2 with a visual representation shown in figure 2. We trained out model using six sensors based on a subset of the SSN tasked with GEO belt monitoring, otherwise known as the Ground-based Electro-Optical Deep Space Surveillance (GEODSS) system [15], extended by a grouping of other SSN sensors (Diego Garcia, Haystack, HEH Naval, Maui, Moron Moss, and Socorro). We then evaluated our model utilizing a further subset of the SSN with up to ten sensors to demonstrate scalability.

Table 2: Sensor Locations

Sensor	Longitude	Latitude	Elevation
Socorro	-106.6599°	33.8172°	1250 m
Maui	-156.2578°	20.7088°	3052 m
Diego Garcia	72.45222°	-7.41173°	0 m
Moron Moss	-5.615833°	37.174722°	87 m
Haystack	-71.488247°	42.623311°	131 m
HEH Naval	114.1656095°	-21.815978°	5 m
Cobra Dane	174.091111°	52.737222°	0 m
Ascension	-14.393889°	-7.969444°	0 m
RTS	167.733333°	8.716667°	0 m
Globus II	31.127800°	70.367100°	0 m

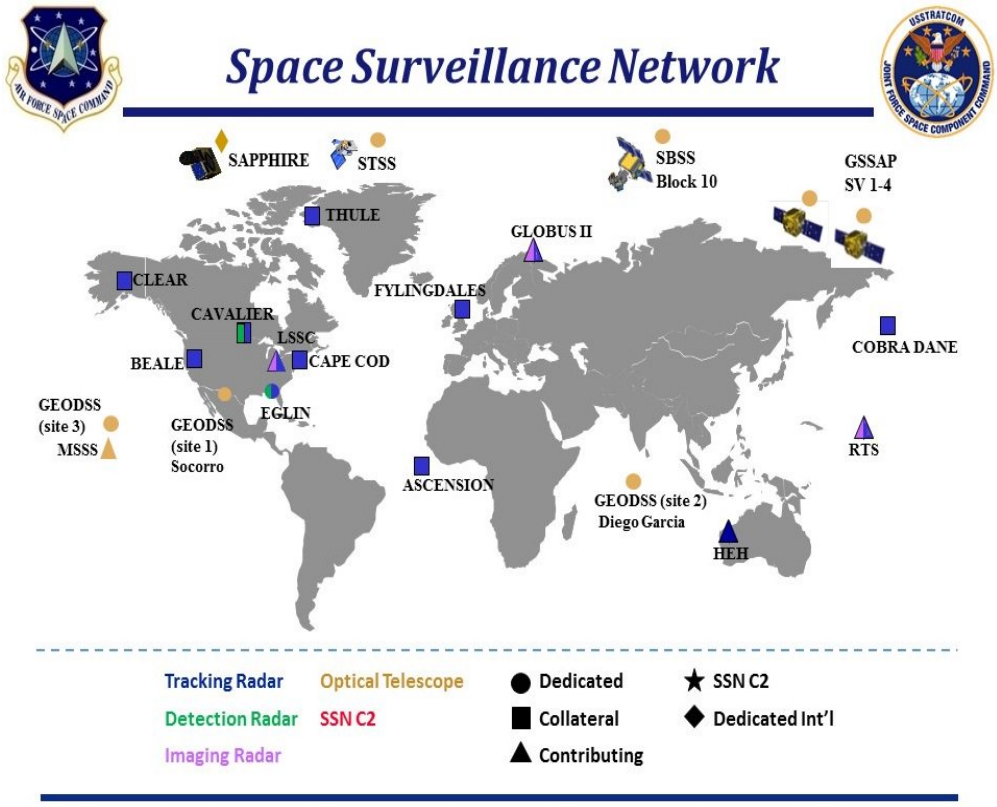


Fig. 2: SSN Locations [15]

2.3 SO orbits

SO orbits were generated using randomly selected orbital elements according to the value ranges in table 3. The generated orbits lie within the GEO regime in an altitude sense, however, they were not limited to the GEO belt and could have polar-like orbits in-terms of inclination. Each randomly generated set of elements is then checked to ensure that its orbit is valid (ie. it doesn't crash into the Earth).

Table 3: SO Orbital Parameters

Parameter	Description	Range
i	Inclination	degrees (0 to 180)
O	Right Ascension of the Ascending Node	degrees (0 to 360)
e	Eccentricity	fractional (0.0001 to 0.7)
o	Argument of Perigee	degrees (0 to 360)
M	Mean Anomaly	degrees (0 to 360)
n	Mean Motion	rev/day

3. MULTI-AGENT REINFORCEMENT LEARNING FORMULATION

Our MARL implementation builds off of previous works using an actor-critic architecture trained with Proximal Policy Optimization (PPO) as a foundation [22]. Recent advances have been made in adapting this formulation for multi-agent scenarios with demonstrated high performance in various cooperation tasks [23]. On top of this foundation, we explore different neural network architectures for the actor and critic using convolutional neural networks (CNN) and fully connected networks (FC). These models are then trained using either population based training (PBT) or population based bandit (PB2) combined with PPO [24], [25].

3.1 Observation Space

In MARL, the observation space refers to the set of all possible observations that each agent can perceive from its environment. It encompasses the information available to an agent, such as states, sensory inputs, or communication messages, which can be used to make decisions and take actions in the environment. In past work, the observation space for the sensor tasking problem was based on a 90x19 grid associated with the azimuth and altitude of a sensor broken into 4° by 4° grid. This grid observation is partitioned into seven regions each associated with the approximate slew time needed to reach that region as shown in figure 3. Each 4° section provides 11 normalized data points listed in table 4.

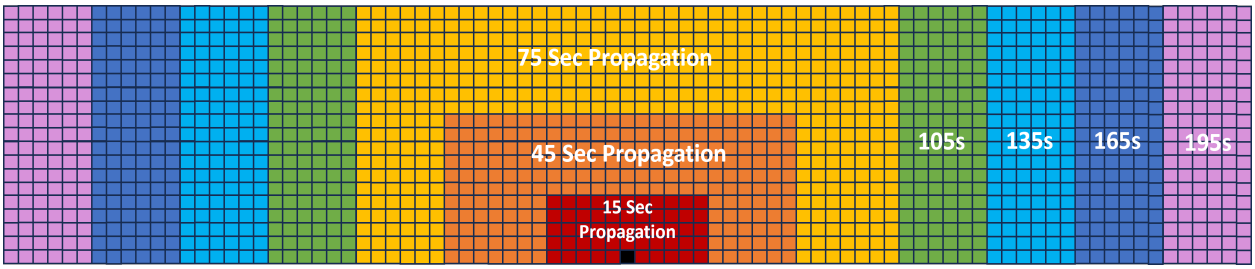


Fig. 3: Single Agent's Observation Grid

We adapted this observation space to our multi-agent environment by providing each agent with their own 90x19 grid. When it is an agent's turn to act, its observation grid is concatenated with the grids of the agents closest to it in the east and to the west. The combination of the these grids make for a total grid space of 270x19, with the same 11 data points used for each 4° by 4° section.

In addition to the 270x19x11 observation grid, the model is also provided with a vector of relative information about the agents to its east and west as shown in table 5.

Because of the asynchronous nature of our environment, only the observations of the acting agents are provided to the model to prevent inactive agents from confusing the model with erroneous information. Typically, this technique is

Table 4: Individual Agent Observation

Layer	Data (Per Observation Grid)
1	Number of SOs
2	Elevation fraction location of SO
3	Azimuth fraction location of SO
4	Range of SO
5	Elevation velocity of SO
6	Azimuth velocity of SO
7	Range velocity of SO
8	Max trace covariance of SOs
9	Sum of SOs trace covariance
10	Mean of SOs trace covariance
11	Current pointing direction

Table 5: Additional Data for East and West Agent Relative to the Acting Agent

Data (Per East and West Agent)
Relative Longitude
Relative Latitude
Relative Altitude
Pointing Azimuth
Time since Last Observation
Time Until Next Action is Completed

used to allow a MARL model to continue updating active agents even when some of the agents may have reached their goal, also known as death masking. In our formulation, we used this technique throughout training so that the model is only updated by observations and reward from the agents who interacted with the environment during each step.

3.2 Action Space

The action space represents the complete set of actions that an agent is capable of performing. In our formulation, the action space is set as the 90x19 grid corresponding to the acting agent's observation grid. An agent then chooses a specific section of the grid to observe, and this is sent back to the environment. We also employed a technique called action masking which discourages agents from attempting to take an action that wouldn't observe an SO (ie. look at empty space). Inclusion of the action masking significantly reduced the amount of training time as agents did not have to spend time discovering where to look to observe an SO.

3.3 Model Architecture

In actor-critic architectures, the actor represents the policy network, which is responsible for selecting actions based on the observations. It takes the current observation as input and produces a probability distribution over possible actions. The critic, on the other hand, approximates the state-action value function. It evaluates the quality of the actor's chosen actions by estimating the expected return or future rewards [26]. This estimate is then compared with the actual received rewards to calculate the prediction error, which is used to update the critic's parameters. The combination of the actor and critic networks allows for continuous learning and policy improvement. The critic provides feedback to the actor by evaluating its chosen actions, while the actor explores and searches for better actions based on this feedback. This interaction between the actor and critic networks leads to the iterative refinement of the policy, resulting in more optimal actions over time.

For our implementation, we explored two different neural network architectures for the actor and critic networks. In the first architecture 9, we used separate CNNs for each of the networks followed by separate FC networks. For the second architecture 10, we used the same CNN for both the actor and critic, with separate FC networks. In both cases, the CNNs process the 270x90x11 grid provided by the environment to represent the acting agents and the agents to their east and west. This output is then concatenated with the data vector containing additional information about the east and west agents relative to the acting agents before being sent to the FC networks associated with the actor and

critic. The action mask is then applied to the output of the actor network before being fed back to the environment. Figure 4 shows an overview of the second architecture.

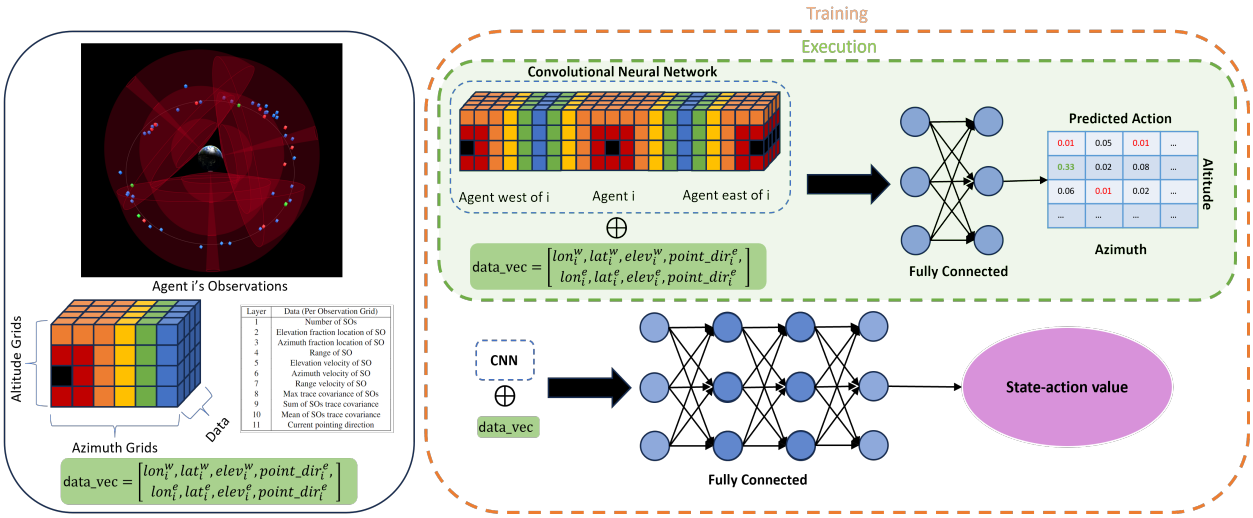


Fig. 4: Model Architecture Using Shared CNN

3.4 Reward

When training MARL agents, the reward function is vital as it acts as an incentive mechanism to guide the agents towards desired states or actions in the environment. A poorly constructed reward function can lead to slow training and subpar performance. In our sensor tasking environment, there are various objectives we could consider, such as minimizing the revisit time of all SOs in the catalog, maximizing the number of unique SOs observed within a defined observation window, or ensuring that uncertainties for any SO stay below a certain threshold. To achieve these objectives, the reward function can be modified to optimize performance based on the desired metric.

We explored a reward function that would properly reward individual agent actions, while encouraging agents to take actions that would benefit not just themselves, but the group as a whole. In the single-agent implementations a reward function based on improvements of the covariance scaled by the slew time needed to perform an action were employed. This reward is shown in Eq 1

$$R = \frac{max(tr(P_{k|k-1}^{(a_k)})) - max(tr(P_{k|k}^{(a_k)}))}{T[k] - T[k-1]} \tag{1}$$

where $tr(P_{k|k-1}^{(a_k)})$ is the trace covariance of the SOs within the agent's FOV after they performed action a_k and $T[k]$ is the time at environment step k. For our multi-agent implementation, we adapted this reward function such that it would provide an individual reward to each agent based on how much it was able to reduce the covariance of the SOs within its own FOV scaled by the slew time needed to perform its action. As will be seen in the results, this reward did not always correlate with larger reductions of the mean trace covariance in the overall environment. Future work will investigate the reward function further in pursuit of one that has a higher correlation to our performance metrics and encourages more cooperation between the agents.

3.5 Training

We explored two training methods for our model architecture; PBT and PB2. In both of these methods, a population of models are trained concurrently with different hyperparameters in what is called the exploration phase, this is then followed by an exploitation phase that finds the best performing models, replicates and mutates those models' hyperparameters across the population and then starts another exploration phase with these new models. The main difference between these two methods is that PBT will incrementally sample the new hyperparameter values within some bounds from the best performing models, while PB2 can instead sample hyperparameter values from the entire search space using a Gaussian distribution centered around the best performing models [25].

Within PB2 and PBT, each individual model uses PPO to update its hyperparameters during the exploration phase. The inclusion of PPO both simplifies and improves the stability of the learning process by preventing an individual model's policy from deviating too far from its previous policy such that it doesn't make large and potentially harmful changes [22].

4. RESULTS AND DISCUSSION

For this paper, we evaluated our MARL model with multiple environment formulations and sensor parameters. In a real-world context, environments are not static with the number of sensors available and number of observable satellites changing frequently. We explored the ability of our models to scale based on these changing environmental parameters even when trained on a relative simple environment and with few other agents. Throughout these experiments, our models are compared against myopic policies acting as baselines for performance.

We used reduction in mean trace covariance of the environment as our main performance metric. Our first experiment consisted of testing our model against the myopic policies in a nominal case with 100 SOs and six sensors to reflect the training environment. From here, we began scaling the number of SOs and sensors in the environment up to a maximum of 1000 SOs and 10 sensors. We generally saw that our models were able to outperform the myopic policies up to 600 SOs when trained in an environment of only 100 SOs and with a number of sensors equivalent or more than the number in the training environment. However, there is a drop in performance beyond 600 SOs and when the number of sensors is below the number used in the training environment. We reason that this is reaching the limitation of the ability of the model to scale and it is worth investigating if a model trained with a larger number of SOs and fewer agents would be able to scale further. Even with this limitation reached, the current catalog of active GEO SOs is around 500, demonstrating that our model can effectively maintain the catalog.

In addition to the reduction in the mean trace covariance, we also measured the mean total reward gained by each agent at the end of an episode. Along this metric, we saw that our models out performed the myopic policy even beyond the 600 SO limitation evident with mean trace covariance. Why our reward does not correlate with the trace covariance performance metric warrants further investigation, but we can surmise that it does not properly capture the complexity of the SSA environment.

4.1 Baseline Myopic Policies

We implemented two models based on those developed in [1] to act as myopic policies for benchmarking our MARL model. The first of these models is a simple greedy model that chooses the section of the 90x19 observation grid that has the largest trace covariance. Our second model is a more advanced greedy model that chooses the section of the grid with the largest trace covariance scaled by the amount of slew time it would take for that agent to view that section. The action of the advanced greedy is given in 2.

$$a^* = \text{argmax}\left(\frac{tr(P_i)}{\delta t_i}\right) \tag{2}$$

where $tr(P_i)$ is the trace covariance of the SOs within grid section i and δt_i is the slew time it would take to view grid section i.

4.2 Nominal Results

For this particular seeded environment, the CNNv1 MARL model achieves a final mean trace covariance of 0.00971, which is an improvement of 3.70% over the advanced greedy policy. Similarly, the CNNv2 MARL model achieves a final mean trace covariance of 0.00968, signifying a 4.01% over the advanced greedy policy.

100 Monte Carlo runs were carried out to get the aggregate performance for the different tasking algorithms. The outcomes were analyzed with respect to the mean episodic reward and the final mean trace covariance, both assessed over 100 seeded rollouts Figure 6 shows the distribution of these two metrics. Although the MARL models are specifically trained to maximize the mean episodic rewards, the CNNv2 MARL model surprisingly received a lower mean episodic reward compared to the advanced greedy policy. However, both MARL models were able to perform better than the myopic policies and arrive at a lower final mean trace covariance. Among the tasking algorithms, the CNNv2 MARL model has the best aggregate performance and achieves an average mean trace covariance of 0.00948, followed closely by the CNNv1 MARL model with a value of 0.00956, advanced greedy policy with 0.00980, and finally the greedy policy with 0.01027.

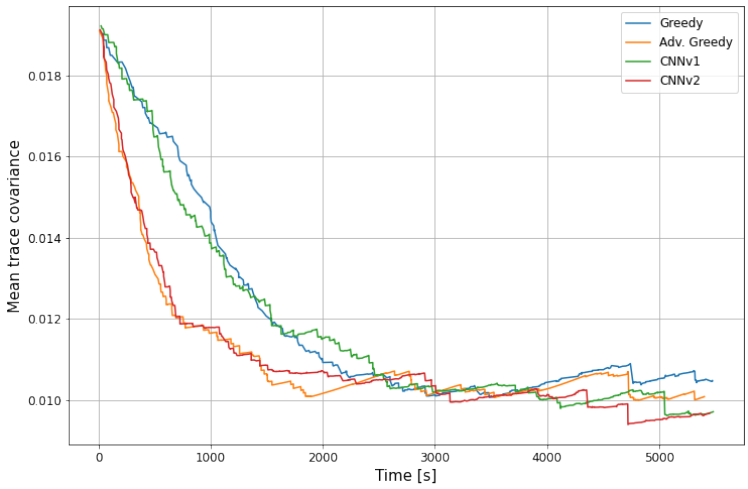
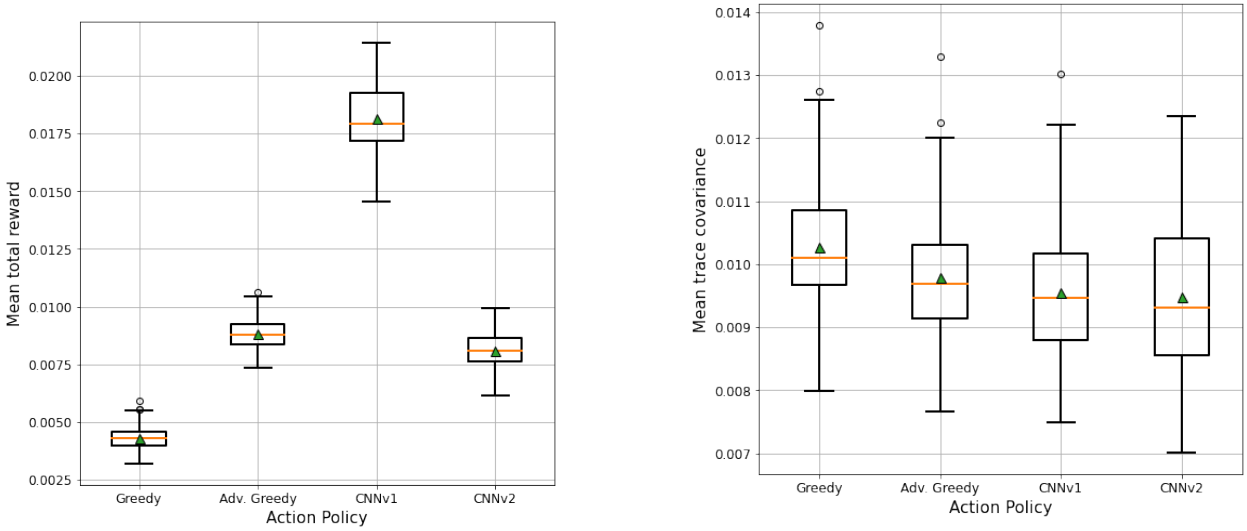


Fig. 5: Evolution of mean trace covariance of all SOs over a single seeded scenario.



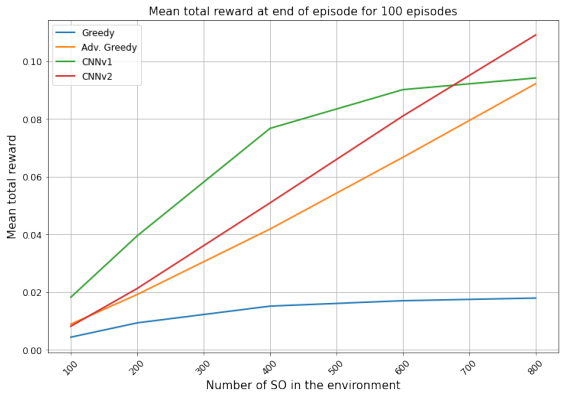
(a) Mean episodic reward for 100 Monte Carlo runs.

(b) Mean trace covariance at the end of episode for 100 Monte Carlo runs.

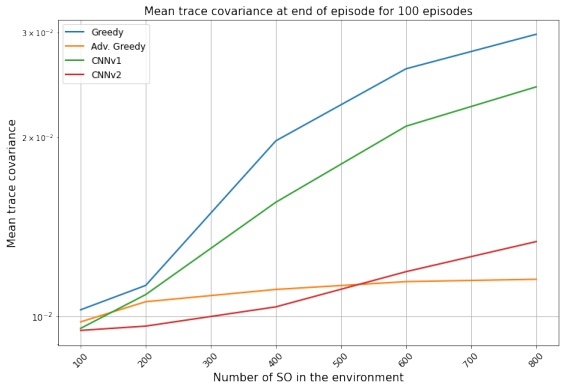
Fig. 6: Aggregate performance of the MARL models in the nominal SSA environment.

4.3 Robustness to Variation in Number of Space Objects

We evaluated the two MARL models and two myopic policies across scenarios involving varying number of space objects. The primary objective of this experiment was to understand how the MARL models will perform when there are more SOs in the scenario than the models saw during training. We evaluated the models at the following numbers of SOs in the environment: 100, 200, 400, 600, 800, 1000. Figures 7a and 7b show the mean episodic reward and the mean trace covariance for each agent at the end of a full episode, for different numbers of SOs in the environment, respectively. Despite being trained with an environment containing only 100 SOs, the CNNv2 model outperforms the advanced greedy policy when there are 200 and 400 SOs in the environment. As the MARL models were trained to maximize the mean episodic reward, they were generally able to achieve a higher mean episode reward compared to the myopic policies across all scenarios. However, a noteworthy observation is the discontinuity between mean episodic reward and final mean trace covariance. While the MARL models consistently excel in the mean episodic reward across all scenarios, this achievement does not uniformly translate to having a lower final mean trace covariance. This highlights the complexity of the SSA sensor tasking problem.



(a) Aggregate mean episodic reward for 100 Monte Carlo runs.



(b) Aggregate mean trace covariance at the end of episode for 100 Monte Carlo runs.

Fig. 7: Aggregate statistics over 100 Monte Carlo runs for scenarios with 6 sensors and different number of SOs.

Table 6 shows the aggregate statistics of the 100 Monte Carlo runs for the difference scenarios, each with a different number of SOs in the environment. The average mean trace covariance at the end of the observation windows are then compared to the advanced greedy policy by taking the percentage difference between the different tasking algorithms and the advanced greedy policy, this is denoted by $\Delta\%$ in the following tables.

Table 6: Variation in the final mean trace covariance as a function of number of SOs over 100 seeded runs

	Covariance of all SOs at the end of Observation Window								
SOs #	**100**			**200**			**400**		
	Mean	**Std**	$\Delta\%$	**Mean**	**Std**	$\Delta\%$	**Mean**	**Std**	$\Delta\%$
Greedy	0.01027	0.00102	-4.78	0.01128	0.00060	-6.50	0.019732	0.00100	-77.63
Advanced Greedy	0.00980	0.00101	-	0.01059	0.00062	-	0.01111	0.00049	-
CNNv1	0.00956	0.00106	2.48	0.01089	0.00058	-2.80	0.01556	0.00093	-40.09
CNNv2	**0.00948**	0.00122	**3.25**	**0.00964**	0.00083	**9.00**	**0.01038**	0.00053	**6.52**

SOs #	**600**			**800**			**1000**		
	Mean	**Std**	$\Delta\%$	**Mean**	**Std**	$\Delta\%$	**Mean**	**Std**	$\Delta\%$
Greedy	0.02607	0.00102	-127.68	0.02979	0.00085	-157.83	0.03216	0.00084	-173.85
Advanced Greedy	**0.01145**	0.00039	-	**0.01155**	0.00034	-	**0.01174**	0.00032	-
CNNv1	0.02088	0.00094	-82.36	0.02431	0.00077	-110.40	0.02681	0.00075	-128.27
CNNv2	0.01190	0.00051	-3.91	0.01337	0.00049	-15.67	0.01478	0.0005	-25.85

4.4 Robustness to Variation in Number of Sensors

We subsequently evaluated the two MARL models and two myopic policies across scenarios involving varying number of sensors. The MARL models were originally trained on an environment with six globally distributed sensors and are evaluated on environments with different numbers of sensors as well as novel sensor locations that were not included as part of the original training setup.

Three different scenarios are considered; three operational sensors from the GEODSS system, five globally distributed sensors, and 10 globally distributed sensors.

For the first scenario, the three operational sensors in the GEODSS system—Diego Garcia, Maui, and Socorro—are used. Table 7 shows the aggregate statistics of the 100 Monte Carlo runs for the different scenarios using these three sensors, each with a different number of SOs in the environment. Again, we are using the advanced greedy policy as our baseline solution. The CNNv1 MARL model struggles to perform in this environment, with their percentage difference reaching close to -48.45%, but it was still able to outperform the greedy policy. On the other hand, the CNNv2 MARL model fared better and only has a percentage difference of -11.90%. This might be caused by the

small overlap between the sensors' FOR and the larger separation between neighboring sensors than those that were used for training the MARL agents.

Table 7: Final mean trace covariance as a function of number of SOs over 100 seeded runs for an environment with 3 sensors

	Covariance of all SOs at the end of Observation Window								
SOs #	**100**			**200**			**400**		
	Mean	**Std**	Δ%	**Mean**	**Std**	Δ%	**Mean**	**Std**	Δ%
Greedy	0.01121	0.00079	-0.26	0.01306	0.00058	-9.46	0.01775	0.00050	-42.42
Advanced Greedy	**0.01118**	0.00090	-	0.01193	0.00060	-	**0.01246**	0.00044	-
CNNv1	0.01184	0.00090	-5.88	0.01374	0.00063	-15.14	0.01668	0.00046	-33.89
CNNv2	0.01148	0.00111	-2.68	**0.01170**	0.00072	1.96	0.01250	0.00048	-0.29

SOs #	**600**			**800**			**1000**		
	Mean	**Std**	Δ%	**Mean**	**Std**	Δ%	**Mean**	**Std**	Δ%
Greedy	0.01992	0.00046	-55.38	0.02109	0.00035	-59.99	0.02175	0.00031	-59.31
Advanced Greedy	**0.01282**	0.00090	-	**0.01318**	0.00037	-	**0.01365**	0.0003	-
CNNv1	0.01853	0.00044	-44.57	0.01953	0.00042	-48.14	0.02027	0.00035	-48.45
CNNv2	0.01355	0.00036	-5.70	0.01449	0.00034	-9.92	0.01528	0.00031	-11.9

For the second scenario, a sensor was removed from the nominal environment to represent sensor maintenance or interruption due to weather. The Socorro site was arbitrary chosen to be removed from the environment. Table 8 shows the aggregate statistics of the 100 Monte Carlo runs for the difference scenarios using the remaining five sensors (Diego Garcia, Haystack, HEH Naval, Maui, and Moron Moss), each with a different number of SOs in the environment. Again, the CNNv1 MARL model struggles to keep up with the advanced greedy policy. However, the CNNv2 MARL model was able to outperform the advanced greedy policy at low SO numbers and its performance degrades as the number of SO in the environment increases. This is due to the MARL models being trained on an environment with only 100 SOs.

Table 8: Final mean trace covariance as a function of number of SOs over 100 seeded runs for an environment with 5 sensors

	Covariance of all SOs at the end of Observation Window								
SOs #	**100**			**200**			**400**		
	Mean	**Std**	Δ%	**Mean**	**Std**	Δ%	**Mean**	**Std**	Δ%
Greedy	0.00997	0.00089	-3.12	0.01102	0.00054	-6.83	0.01865	0.00082	-72.78
Advanced Greedy	0.00966	0.00092	-	0.01032	0.00053	-	**0.01079**	0.00042	-
CNNv1	0.00975	0.00086	-0.90	0.01137	0.00066	-10.15	0.01621	0.00075	-50.24
CNNv2	**0.00945**	0.00097	2.21	**0.00976**	0.00073	5.38	0.01085	0.00056	-0.49

SOs #	**600**			**800**			**1000**		
	Mean	**Std**	Δ%	**Mean**	**Std**	Δ%	**Mean**	**Std**	Δ%
Greedy	0.02367	0.00069	-114.15	0.02639	0.00064	-134.98	0.02801	0.00055	-143.69
Advanced Greedy	**0.01105**	0.00037	-	**0.01123**	0.00030	-	**0.0115**	0.00026	-
CNNv1	0.02023	0.00064	-82.99	0.02283	0.00066	-103.32	0.02452	0.00055	-113.29
CNNv2	0.01244	0.00043	-12.52	0.01386	0.00045	-23.46	0.01517	0.00039	-32.01

For the third scenario, four additional sensors are added to the nominal SSA environment for a total of 10 sensors. The sensor locations are outlined in Table 2. Table 9 shows the aggregate statistics of the 100 Monte Carlo runs for the difference scenarios using all 10 sensors, each with a different number of SOs in the environment. The MARL agents are able to perform on par to the advanced greedy policy for low number of SOs. However, they struggle to perform in environments with more than 400 SOs.

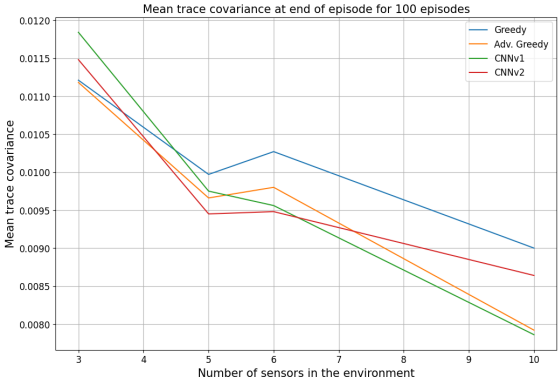
Figure 8 shows the performance of the various tasking strategies on the different scenarios with 100 and 400 SOs. For

Table 9: Final mean trace covariance as a function of number of SOs over 100 seeded runs for an environment with 10 sensors

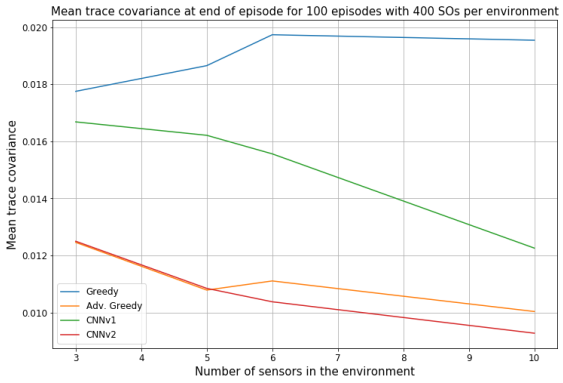
	Covariance of all SOs at the end of Observation Window								
SOs #	**100**			**200**			**400**		
	Mean	**Std**	Δ%	**Mean**	**Std**	Δ%	**Mean**	**Std**	Δ%
Greedy	0.009	0.00124	-13.65	0.01039	0.00086	-14.13	0.01954	0.00175	-94.66
Advanced Greedy	0.00792	0.00119	-	0.0091	0.00083	-	0.01004	0.00068	-
CNNv1	**0.00786**	0.00122	0.76	0.00905	0.00091	0.56	0.01226	0.00087	-22.09
CNNv2	0.00864	0.00176	-9.11	**0.00866**	0.00135	4.87	**0.00928**	0.00087	7.57

SOs #	**600**			**800**			**1000**		
	Mean	**Std**	Δ%	**Mean**	**Std**	Δ%	**Mean**	**Std**	Δ%
Greedy	0.03298	0.00215	-211.85	0.0379	0.00076	-244.26	0.03819	0.00101	-240.55
Advanced Greedy	**0.01058**	0.00053	-	**0.01101**	0.0005	-	**0.01121**	0.00048	-
CNNv1	0.01944	0.00163	-83.75	0.02773	0.00186	-151.94	0.03446	0.0017	-207.28
CNNv2	0.0111	0.00084	-4.94	0.01307	0.00085	-18.746	0.01512	0.00082	-34.81

the 100 SO case, the advanced greedy policy has the best performance in the the three agent scenario, however, as the number of agents increases, the advanced greedy policy starts to be overshadowed by the MARL models. Between the two MARL models, the CNNv2 MARL model initially has a better performance compared to the CNNv1 MARL model, however, it performed worse than the CNNv1 MARL model for the 10 agent scenario. On the other hand, for the 400 SOs case, the CNNv1 MARL model was not able to adapt to the higher number of SOs in the environment and performed subpar when compared to the CNNv2 MARL model and the advanced greedy policy. The CNNv2 MARL model performed on par with the advanced greedy policy at low number of SOs, and outperformed the advanced greedy policy at higher number of SOs.



(a) Performance Versus Number of Sensors With 100 SOs in the environment



(b) Performance Versus Number of Sensors With 400 SOs in the environment

Fig. 8: Performance Versus Number of Sensors

5. CONCLUSIONS

With the proliferation of space objects in Earth's orbit, the curation and maintenance of the SO catalog is becoming increasingly challenging. Novel and efficient sensor tasking algorithms are required to safeguard our space assets and ensure a sustainable space environment. In this work, we explored the usage of DRL agents to efficiently task a network of ground-based sensors in a decentralized fashion. Each agent is only provided with the information of their closest neighboring agents and are required to work collaboratively in the SSA catalog maintenance task. Our training environment consisted of six agents based on a subset of the SSN tasked with reducing the mean trace covariance of 100 SOs.

We developed two models using an actor-critic framework as a foundation, with varying formulations of CNNs and FC networks built on top. These were then trained using either PBT or PB2 combined with PPO. We then evaluated our models based on their ability to reduce the mean trace covariance of an environment. This evaluation started by showing that our models performed better than myopic greedy and advanced greedy policies in a nominal environment. We then expanded our experiments to investigate the robustness of our models against differing numbers of SOs and sensors. When increasing the number of SOs beyond the 100 the models were trained on, we saw that our CNNv2 model was able to perform better than the myopic policies up to 600 SOs, which matches the number of active SOs currently within the GEO catalog. In the second set of experiments, we started by reducing the number of sensors and then gradually increased the number acting in the environment until there were 10 sensors total. We saw that our models struggled to perform better than the advanced greedy policy when the number of sensors was reduced below the training environment, however we saw that their performance improved beyond that of advanced greedy when the number of sensors was increased to match and exceed the training environment.

Overall, we were able to demonstrate that MARL models are an effective method toward a solution to the multi-agent sensor tasking problem. However, we did see a drop in performance when we evaluated our models with higher numbers of SOs. This and the lack of a strong correlation between the reward and our main performance metric warrants further investigation. Beyond these investigations, we would like to evaluate other model architectures against our current model.

One of these proposed future works include exploring the usage of Graph Neural Networks (GNNs) for decentralized SSA sensor tasking. More recently, GNNs paired with MARL have shown sustained (and in some cases improved) performance even when information is limited and agents must rely only on their local observations. This performance has also shown to be scalable even when the MARL algorithms are trained using relatively few agents.

Finally, we plan to test these methods with real world sensor networks to evaluate their effectiveness at catalog maintenance and eventually in more complex scenarios such as those that contain maneuvering satellites.

ACKNOWLEDGEMENT

Research was sponsored by the United States Air Force Research Laboratory and the Department of the Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Department of the Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. This work is partially supported by the Aerospace Corporation's University Partnership Program. The authors also thank the MIT SuperCloud and Lincoln Laboratory Supercomputing Center for providing HPC, database, and consultation resources that have contributed to the research results reported in this paper.

REFERENCES

[1] Peng Mun Siew and Richard Linares. Optimal tasking of ground-based sensors for space situational awareness using deep reinforcement learning. *Sensors*, 22(20):7847, 2022.

[2] Siddharth Nayak, Kenneth Choi, Wenqi Ding, Sydney Dolan, Karthik Gopalakrishnan, and Hamsa Balakrishnan. Scalable multi-agent reinforcement learning through intelligent information aggregation. In *International Conference on Machine Learning*, pages 25817–25833. PMLR, 2023.

[3] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[4] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[5] Sydney Dolan, Siddharth Nayak, and Hamsa Balakrishnan. Transfer learning for space traffic management. *arXiv preprint arXiv:2211.03658*, 2022.

[6] D Jang, PM Siew, G Gondelach, and R Linares. Space situational awareness tasking for narrow field of view sensors: A deep reinforcement learning approach. In *71st International Astronautical Congress. International*

Astronautical Federation, the International Academy of Astronautics, and the International Institute of Space Law, Virtual, 2020.

[7] Peng Mun Siew, Daniel Jang, Thomas G Roberts, Richard Linares, and Justin Fletcher. Cislunar space situational awareness sensor tasking using deep reinforcement learning agents. 2022.

[8] Peng Mun Siew, Daniel Jang, Thomas G Roberts, and Richard Linares. Space-based sensor tasking using deep reinforcement learning. *The Journal of the Astronautical Sciences*, 69(6):1855–1892, 2022.

[9] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[10] Saber Astronautics. Space Cockpit | Saber Astronautics.

[11] Brandon Craig Rhodes. Pyephem: astronomical ephemeris for python. *Astrophysics Source Code Library*, pages ascl–1112, 2011.

[12] David Vallado and Paul Crawford. Sgp4 orbit determination. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, page 6770, 2008.

[13] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter. *Kalman filtering and neural networks*, pages 221–280, 2001.

[14] Johannes Herzog. *Cataloguing of objects on high and intermediate altitude orbits*. PhD thesis, Universität Bern, 2013.

[15] 18th Space Defense Squadron (@18thSDS) / X, February 2018.

[16] Space Surveillance Sensors: GEODSS (Ground-based Electro-Optical Deep Space Surveillance) System (August 20, 2012), August 2012.

[17] Space Surveillance Sensors: Globus II Radar (June 1, 2012), June 2012.

[18] Timothy D Hall, Gary F Duff, and Linda J Maciel. The Space Mission at Kwajalein.

[19] Dipl-Ing (FH) Christian Wolff. AN/FPS-108 "Cobra Dane" - Radartutorial. Publisher: Dipl.-Ing. (FH) Christian Wolff.

[20] S M Lederer, E G Stansbery, H M Cowardin, P Hickson, L F Pace, K J Abercromby, and P W Kervin. The NASA Meter Class Autonomous Telescope: Ascension Island.

[21] U.S. Space Surveillance Telescope in Australia achieves initial operational capability, September 2022.

[22] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[23] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

[24] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population Based Training of Neural Networks, November 2017. arXiv:1711.09846 [cs].

[25] Jack Parker-Holder, Vu Nguyen, and Stephen Roberts. Provably Efficient Online Hyperparameter Optimization with Population-Based Bandits, June 2021. arXiv:2002.02518 [cs, stat].

[26] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

APPENDIX

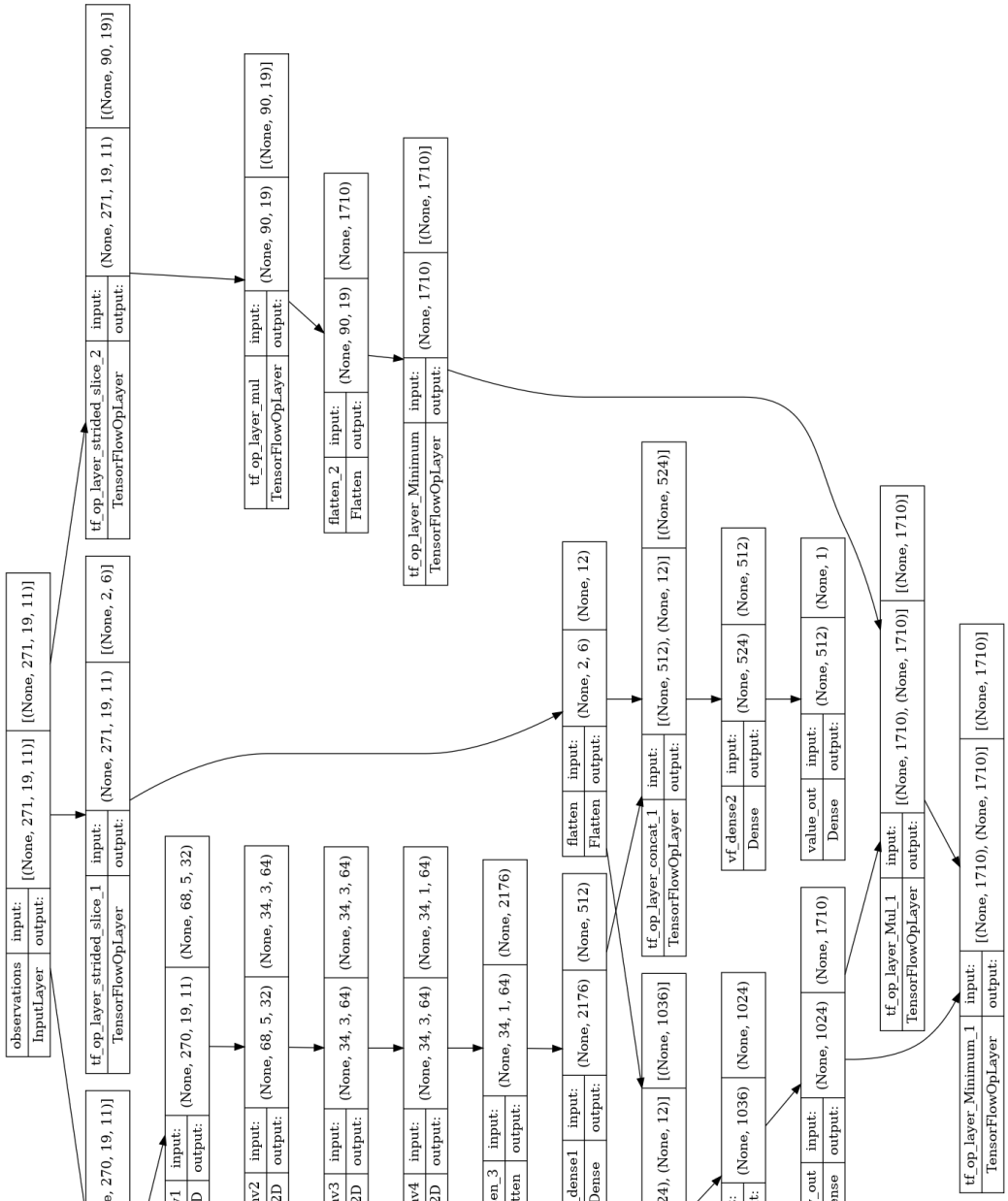


Fig. 9: Model Architecture for CNNv1

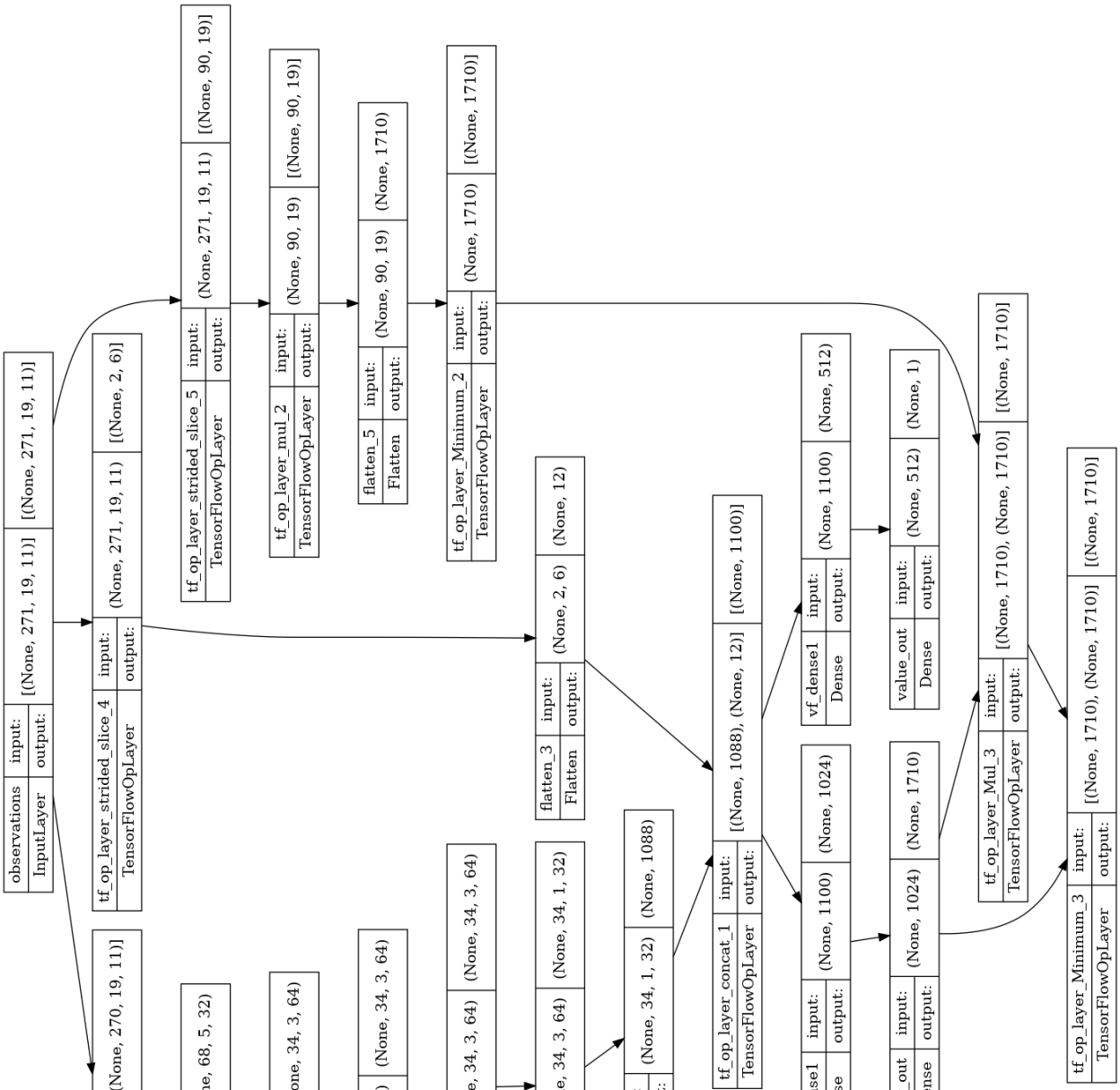


Fig. 10: Model Architecture for CNNv2 trained with Population Based Training