aerospace

MDPI

Article

A New Strategy of Satellite Autonomy with Machine Learning for Efficient Resource Utilization of a Standard Performance CubeSat

Desalegn Abebaw Zeleke [1] and Hae-Dong Kim [2,*]

[1] University of Science and Technology (UST), Daejeon 500101, Republic of Korea
[2] Department of Aerospace and Software Engineering, Gyeongsang National University, Jinju 13557, Republic of Korea
* Correspondence: haedkim@gnu.ac.kr; Tel.: +82-10-3401-7655

Abstract: A mega constellation of Nano/microsatellites is the contemporary solution for global-level Earth observation demands. However, as most of the images taken by Earth-observing satellites are covered by clouds, storing and downlinking these images results in inefficient utilization of scarce onboard resources and bandwidth. In addition, the trend of making satellite task execution plans by ground operators demands the efforts of experts or simulators to predict the real-time situation of satellites and to decide which tasks should be executed next. Granting controlled autonomy to satellites to perform onboard tasks will boost mission effectiveness. We experimented with granting controlled autonomy for satellites in performing onboard image classification and task scheduling. We designed a convolutional neural network-based binary image classification model with more than 99% accuracy in classifying clear and cloudy images. The model is configured to perform inference in low-performance computers of ordinary Cubesats. Moreover, we designed an autonomous satellite task scheduling mechanism based on reinforcement learning. It performs better than a custom heuristic-based method in scheduling onboard tasks. As a result, the proposed classification and scheduling techniques with machine learning ensured efficient utilization of onboard memory, power, and bandwidth in the highly resource-constrained CubeSat platforms and mission accomplishment of Nano/microsatellite constellations.

Keywords: autonomous cubesat; onboard image classification; onboard scheduling; onboard resource utilization; Q-learning

Citation: Zeleke, D.A.; Kim, H.-D. A New Strategy of Satellite Autonomy with Machine Learning for Efficient Resource Utilization of a Standard Performance CubeSat. *Aerospace* **2023**, *10*, 78. https://doi.org/10.3390/aerospace10010078

Academic Editors: Paolo Tortora and Angelo Cervone

Received: 16 November 2022
Revised: 15 December 2022
Accepted: 22 December 2022
Published: 13 January 2023

1. Introduction

Cubesats are becoming increasingly utilized for various space missions ranging from technology demonstration to scientific and commercial missions. However, they are relatively constrained in size, power, and weight. These constraints demand the miniaturization of radiation-tolerant onboard computers and power units. The way mission operations are planned and executed by satellites should take into account the efficient utilization of these scarce onboard resources.

Most of the satellites operating in space today are largely ground-dependent for control commands and mission operation schedules. These satellites have no autonomy to decide the next best task based on their onboard resource and space environment. Additionally, due to the uncertainty of the space environment and the difficulty of fixing errors onboard, satellites may not be expected to become fully autonomous, as is the case with industrial robots. However, satellite autonomy could be designed by granting rights to the satellite to independently execute tasks such as operation scheduling, image processing, event detection, and fault detection. This study focuses on experimenting with onboard image processing tasks and autonomous mission planning at the same time.

Machine learning technologies such as artificial neural networks are the most utilized approaches in designing autonomous machines. The application of artificial neural networks in autonomous spacecraft mission planning requires a large amount of previously performed planning data to train the machine so that it can learn how to make future decisions. Therefore, this could be difficult for CubeSat missions with no heritage mission data to be used as a training dataset [1].

A study [2] proposes the importance of increasing nanosatellite autonomy in the Asteroid Impact Mission (AIM). In this study, a CubeSat applies artificial neural network algorithms to learn and identify the asteroid impact event onboard as it stays in orbit a month before the actual asteroid impact event occurs. The CubeSat, during its stay before the event date, will take pictures of the asteroid and artificially make after-impact image classes to construct training datasets. In this case, the CubeSat has the autonomy of deciding by itself, but it needs to have a high-performance onboard computer and large memory capacity to construct an artificial neural network-based classifier model which will learn from training datasets and be able to detect when the event occurs.

In another study [3] about the Intelligent Payload Experiment (IPEX) CubeSat, the CubeSat is designed to have onboard autonomy on its experimental mission by utilizing several artificial intelligence technologies to perform image processing tasks onboard. In this case, the satellite is designed to make onboard decisions autonomously without the support of ground technicians.

European Space Agency (ESA) has recently performed an experimental program, FSSCat [1], to evaluate the benefit of artificial intelligence in Earth observation. PhiSat-1, launched in September 2020, is one of the two 6-U CubeSats in this ESA mission. PhiSat-1 is an effort to demonstrate the application of onboard image classification algorithms to autonomously classify cloudy images onboard the satellite with the help of Intel's AI-enabled chip. For onboard image classification, PhiSat-1 has used intel's commercial off-the-shelf, low-power, high-performance computer vision chip named Intel® Movidius™ Myriad™ 2 Vision Processing Unit (VPU) to enable the CubeSat to identify cloudy images and discard them onboard. The Deep Neural Network algorithm is also designed to train itself with new images acquired onboard. The team claimed that applying onboard image classification will save about 30% of the satellite bandwidth, as more than two-thirds of the time the Earth is covered with clouds. However, as our research is based on onboard computers of standard CubeSats, such onboard training of the image classifier model is not possible.

In another research about onboard image processing for small satellites [4], the researchers first applied edge detection and sliding window techniques to detect objects on the open ocean and then applied a deep convolutional neural network to classify objects into binary classes as ships or not-ships. However, they used GPU-powered Nvidia's Jetson TX2 embedded system.

As a study [5] suggests, apart from the onboard memory management, efficient usage of onboard power supply has a great influence on the overall performance of nanosatellite constellations as the inter-satellite and satellite-to-ground station communications are power-consuming. In this regard, the proper application of artificial intelligence technologies could enable nanosatellites to make autonomous decisions on board and boost their capability while ensuring efficient utilization of scarce onboard resources [6,7].

2. Background

Scheduling of Earth observation satellites' imaging tasks is generally regarded as an optimization problem and many researchers proposed the implementation of different approaches for solving the problem. Liang et.al [8] proposed a heuristic algorithm for subsystem-level onboard task scheduling, which is designed by considering the precedence of onboard tasks and resources needed by target subsystems executing these tasks and reported that the algorithm generated satisfactory scheduling results in a shorter period. In [9,10] studies, customized heuristic algorithms are designed to support the objective

function and competent results are reported with shorter computation time. In [9], the heuristic is targeted to maximize imaging tasks of an agile Earth observation satellite by considering pitch maneuverability. In [10], the study utilizes a Synthetic Aperture Radar (SAR) sensor, and the heuristic is designed to select observation targets and the best observation time of each target.

For ensuring the autonomy of satellites in performing onboard duties, recent studies have employed artificial intelligence algorithms in onboard task scheduling. In [11], researchers proposed Long-Short Term Memory (LSTM), a deep learning-based planning method to solve a satellite's onboard observation task planning problem and reported acceptable results with reasonable response time. Li et.al [12] proposed stable real-time scheduling using Artificial Neural Network (ANN) algorithms for onboard tasks executed by different subsystems of the satellite. Li et.al [13] studied an online scheduling problem for a single autonomous Earth observation satellite in which they proposed a hybrid heuristic-based online scheduling mechanism with revision and progressive techniques to schedule normal and urgent tasks appearing in the scheduling horizon. Yao et.al [14] proposed an autonomous task-planning mechanism for individual satellites which are members of a multi-autonomous satellite constellation based on an improved branch-and-bound strategy focusing on the availability of onboard storage.

However, most of the reviewed research only considered planning of next imaging tasks based on immediate optimality without considering the contribution of the selected task in bringing overall scheduling optimality. The studies considered imaging and downlinking tasks in different satellite orbits and maneuverability within short and long planning horizons. Additionally, separate studies in [1–3] deal with onboard image classification to enhance the autonomy of satellites. In all cases, the onboard image classification is not yet considered an onboard task such as imaging or downlinking and is included in the onboard scheduling.

In this study, we proposed a new strategy of satellite autonomy with machine learning for efficient resource utilization, which is considering the effect of including onboard image classification on the effectiveness of upcoming imaging and downlinking tasks as well as overall resource utilization. Indeed, we assumed that a CubeSat has the standard performance onboard computer and general memory. For this purpose, we designed a lightweight convolutional neural network (CNN) image classification algorithm to discard cloudy images onboard and then we designed an onboard task scheduling algorithm using Reinforcement Learning (RL). This algorithm will enable nanosatellites to evaluate available onboard tasks, including autonomous image classification, and make optimal execution schedules ensuring efficient utilization of onboard resources.

Comparing the performances of our method with algorithms employed in different studies becomes difficult as the basis of the problem, constraints, and mission objectives are mostly not similar to our case [15]. Hence, for the sake of evaluating the efficiency of our method, we have designed a task scheduling based on a heuristic algorithm and compared its results with the RL-based scheduling in light of efficient resource utilization, as well as the number of scheduled imaging and downloading tasks in the planning horizon.

The main contributions of this research work are:

- Design an onboard image classification model that can run on a standard CubeSat computer to autonomously classify cloudy and clear images so that the satellite can ensure efficient use of bandwidth and onboard storage while discarding cloudy images onboard.
- Formulate the task scheduling problem in the form of a Markov Decision Process (MDP) and ensure the next task in the schedule is not only locally optimal but also contributes to bringing an overall optimal scheduling solution.
- Design an autonomous task scheduling based on a reinforcement learning algorithm for a CubeSat to make schedules of its imaging, downlinking, pointing, and onboard image classification tasks. In addition, assess the importance of including onboard image classification in the schedule.

- Compare the efficiency of the designed scheduling algorithm against a heuristic-based scheduling algorithm in light of the scheduling objective and overall resource utilization.

3. Problem Statement

Advanced High-REsolution Image and Video satellite (A-HiREV) is a 6U CubeSat designed by the Korean Aerospace Research Institute (KARI). It is an advanced version of HiREV CubeSat [16]. It is initiated for an Earth observation mission with an optical camera that has a 5M ground resolution. The satellite is currently in the test phase and is expected to be launched into ISS or Sun Synchronous Orbit. In this study, we assumed a constellation of A-HiREV-type CubeSats with a standard onboard processor.

A-HiREV is equipped with a UHF/VHF antenna for the transmission of telemetry data and the reception of commands from the Daejeon ground control station. For research purposes, we assumed an additional ground station in East Africa, Ethiopia. For downlinking high-resolution payload data, the satellite has an S-band transmitter with a 0.25 MB/sec transmission rate. It also has an onboard battery with a capacity of 40 watt-hours (wh) and an 866 MHz Xilinx Zinq-7020 series onboard computer that has a 256 MB RAM and 16 GB external memory. Figure 1 shows HiREV's structural design [16] which is identical to the ongoing A-HiREV's design.

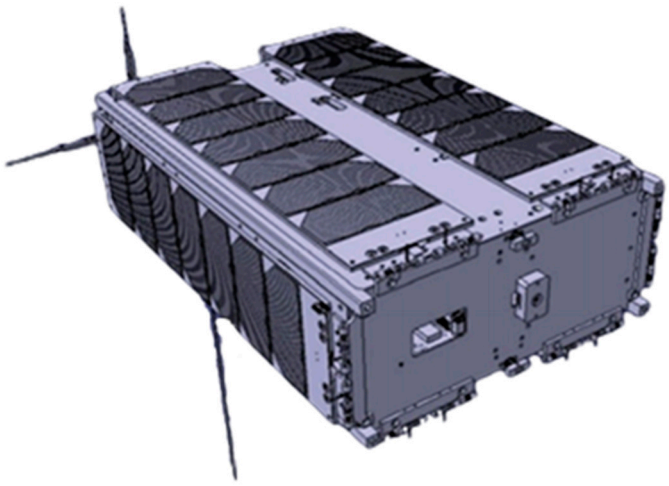


Figure 1. Structural Design of HiREV.

Similar to other Earth observation satellites with an optical payload, A-HiREV could take images of selected locations only during the daytime. In addition, the nature of the payload does not support viewing of the Earth's surface through dense clouds. According to studies [17–19], more than 66% of the time, the Earth's surface is covered by clouds. This implies that more than half of the images taken by A-HiREV are not clear images of the Earth's surface. The payload is capable of taking an image of resolution 3376 $\times$ 2704 pixels; a single uncompressed image takes 27.4 MB of onboard storage and needs 109.6 s to downlink this image with its S-band transmitter to one of the two ground stations. Considering the mega constellation of A-HiREV satellites, control of satellite operations from ground experts will be difficult and the data generated from the camera will result in a shortage of onboard storage since the downlink rate is too slow to send acquired images compared to image acquisition and storing rate.

In the usual operation scenario, it is expected that operators from the ground control stations send a schedule of operational tasks for the satellite to perform sequentially. This schedule is made by ground operators every day or every two days and uploaded to the satellite. However, this technique does not allow the satellite to autonomously make decisions considering the current situation of its subsystems, operational conditions, and available onboard resources.

4. Proposed Methodology

In this study, assuming A-HiREV as part of the mega constellation of similar satellites, we proposed granting controlled autonomy to satellites in performing certain onboard tasks. It results in efficient utilization of highly constrained onboard resources as well as improvement of the overall mission performance of individual satellites and their constellations. For this purpose, we experimented with the effects of granting autonomy in terms of onboard image classification and autonomous task scheduling.

4.1. Onboard Image Classification Using Deep Learning

This method enables the satellite to classify acquired images into cloudy and relatively clear images. This will be achieved by equipping the satellite with a simplified deep learning model which is trained to make image classification. This approach helps the satellite make autonomous decisions for identifying useful and useless images and discard cloudy ones onboard. As a result, it enables the satellite to have efficient utilization of crucial onboard resources such as memory, power, and communication bandwidth as it discards images covered with dense clouds and refrains from downlinking them.

CNNs are becoming state-of-the-art techniques for image classification tasks applied in various disciplines. The technology is not fully utilized in space applications [20] as it usually needs huge processing capacity, and such computing devices are not yet manufactured as space-grade. However, within the capacity of A-HiREV's onboard computer, we can use simplified deep-learning models for onboard image classification.

As image classification by itself is a resource-demanding task, we decided to build a custom CNN classifier model and train it on the ground. The model is then pruned and quantized to fit into A-HiREV's onboard computer for inference purposes. These techniques reduce the computational resource demand and memory footprint of the model during the inference phase by eliminating unnecessary parameters [21–23].

We deployed and tested the quantized CNN classifier model on the ZedBoard platform which has similar specifications and performance as the actual onboard computer used in A-HiREV. The Deep Neural Network Development Kit (DNNDK) is a deep learning environment developed for Zynq-7000. We followed detailed setup and implementation steps as described in the DNNDK user guide [24].

4.2. Autonomous Task Scheduling

The second method is to allow the satellite to receive operation commands from ground control stations and autonomously make an optimal schedule for the order of execution considering its onboard resources. The tasks identified for this purpose are nadir pointing for image acquisition, taking images of selected locations, ground pointing for communication with the ground station, downloading acquired images to ground stations, pointing towards the sun to maximize battery charging, and most importantly, onboard image classification tasks.

A satellite is a system of subsystems operating together for the success of a common mission. Every subsystem plays its role in a synchronized manner. Individual tasks executed by the satellite need to be lined up in a certain order of execution. This order of execution is called a task schedule or plan. The importance of making a schedule is to effectively achieve the objective set by ground operators while keeping the health of the satellite and its subsystems.

Satellite task scheduling is an optimization problem that tries to make an optimal order of execution in a given scenario within an environment, having observed and assumed constraints. There are different approaches to making an optimal satellite task schedule. Rule-based scheduling is a straightforward expert-led solution for making execution orders for satellite tasks. But this method greatly needs expert knowledge to make complex rules of execution. It is also less scalable for additional tasks and optimality is not always guaranteed [8]. A genetic algorithm (GA) is another alternative to deal with complex problems such as task scheduling. There is plenty of satellite task scheduling research

based on genetic algorithms [25,26]. Mixed-Integer Linear Programming (MILP) is also another approach to making satellite task schedules [27–29]. These days, with the recent advancement of machine learning, people have started to use reinforcement learning algorithms for task scheduling [30,31].

4.2.1. Markov Decision Process (MDP)

A Markov Decision Process (MDP) is a general framework for modeling and solving decision-making problems. MDPs formally described an environment for reinforcement learning (RL). Figure 2 shows the agent–environment interaction in a general Markov Decision Process. As illustrated, the agent (the satellite) chooses an action (At) in the current state (St), receives a reward (Rt + 1) from the environment, and then transits to the next state (St + 1). The process is repeated for available states in the planning horizon. The objective is to find a policy that maximizes the expected reward.

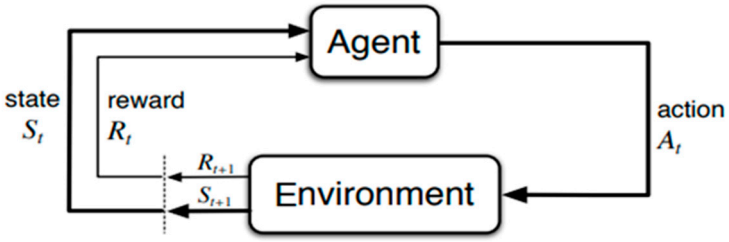


Figure 2. Agent–Environment Interaction in Markov Decision Process.

MDP is formally defined in four tuples: the state space (S), action space (A), transition function (T), and reward function (R). M = (S,A,T,R). MDP formulation of the satellite task scheduling problem is explained in Section 4.

4.2.2. Reinforcement Learning (RL)

Reinforcement learning is one of the basic machine learning paradigms, besides supervised, semi-supervised, and unsupervised learning. What is practiced in supervised learning is teaching a machine by a collection of approved training sets and letting it make its own decision for new cases based on what it has learned from the training set. The image classification technique discussed above is an example of a supervised machine learning methodology. Whereas in reinforcement learning, the agent (satellite) will be in a certain situation (state) with a possible set of actions it can take. The decision to take any of the actions and transiting to the next state is positively or negatively rewarded with reward values computed in terms of the action's role in the optimal achievement of the planning objective. Then the agent learns to take the best actions that will maximize its total reward (or minimize its cost in our case).

Q-learning is a temporal difference control algorithm. It is one of the approaches to implementing reinforcement learning, given the MDP requirements are satisfied. The next section holds details of how Q-learning is implemented.

5. Implementation

5.1. Onboard Image Classification using Deep Learning

Generally, for an image classification task, we need to build a classifier model that learns how to classify images and then use it for inference. For the classifier to learn how to classify, we need to prepare a dataset of initially labeled classes of images.

5.1.1. Preparing Dataset

Image classification is a supervised machine learning technique of teaching the machine with collected sets of images in different classes and designing an algorithm to study the details of images in every class and learn what makes every image a member of a

certain class. Therefore, preparing a huge collection of satellite images is mandatory for training the classifier.

For this purpose, we have utilized an open release of the Sentinel-2 satellite images dataset used in [32] and released them on Harvard Dataverse on January 2020. In the preprocessing step, we first prepared our dataset by individually selecting clear and cloudy images from the released dataset. The images in the two classes have the following properties:

Clear: images showing a clear view of different parts of the Earth's surface and are useful for Earth observation.

Cloudy: images with dense/light cloud cover and showing no clear view of the Earth's surface but mostly cloudy parts. Images in this class are not useful for Earth observation.

We then resized images to similar dimensions of 256 × 256 pixels for a smooth operation of the classification algorithm and then converted them into an array containing each pixel's color value. When the preprocessing is completed, the next step is building a classifier model to learn the features of images in both classes.

5.1.2. Building Classification Model

After preprocessing collected images and making the dataset, we built a CNN model to learn from the given dataset and be able to predict future unseen images into one of the two classes. We used the TensorFlow platform and Keras machine learning library to build the CNN-based image classifier model. We used a total of 19,200 color images of 256 by 256 pixels size of which 12,800 are used for training, 3200 for validating, and 3200 for testing the classifier model. The validation dataset is a set of randomly chosen 20% images from the training dataset. The random selection is executed in every round of the learning phase to make new validation set in each round. Unlike the validation dataset, the test dataset is a separate set of images that are never used in the learning process.

After experimenting with the model building by changing different parameters of the model and observing classification accuracies, we choose the binary classifier model with convolutional layers followed by max-pooling after each convolutional layer, and one fully connected dense layer at the end. As seen in Table 1, the first convolutional layer has 32 kernels of size five by five. Then max pooling is applied before the next convolutional layer. The second layer has 64 kernels of size five followed by a max pooling before it is flattened and forwarded to the output dense layer. In both convolutional layers, we applied strides of one (or same) and rectified linear unit (relu) for the activation function.

Table 1. Summary of Image Classifier Model.

Layer (Type)	Output Shape	Param #
conv2d_10 (Conv2D)	(None, 256, 256, 32)	2432
max_pooling2d_10 (MaxPooling)	(None, 128, 128, 32)	0
conv2d_11 (Conv2D)	(None, 128, 128, 64)	51,264
max_pooling2d_11 (MaxPooling)	(None, 64, 64, 64)	0
flatten_5 (Flatten)	(None, 262144)	0
dense_5 (Dense)	(None, 2)	524,290
	Total params: 577,986	
	Trainable params: 577,986	
	Non-trainable params: 0	

Figure 3 shows the training and validation accuracy of the classifier when running in 20 epochs (learning rounds). As seen in the figure, the binary classifier, on average, has over 99% training and validation accuracies. Figure 4 shows the training and validation losses of the classifier in each learning round and Figure 5 shows a sample run of the classifier displaying the predicted classes versus actual image classes when tested to classify 18 images from the testing dataset. The labels at the top of the sample images show predicted image class and actual image class (in square bracket) names, respectively.

While testing the classifier model, before going through the classification process, test images of any size have to be resized into similar sizes (256 × 256) and also con-verted into an array of color codes of pixels as images used in the training set.

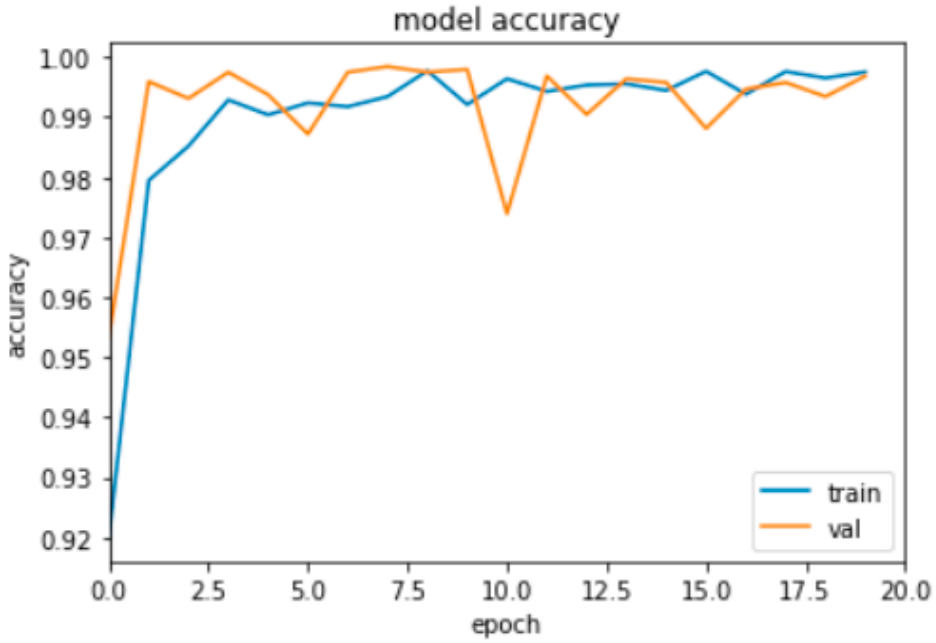


Figure 3. Training and Validation Accuracy of Image Classifier Model.

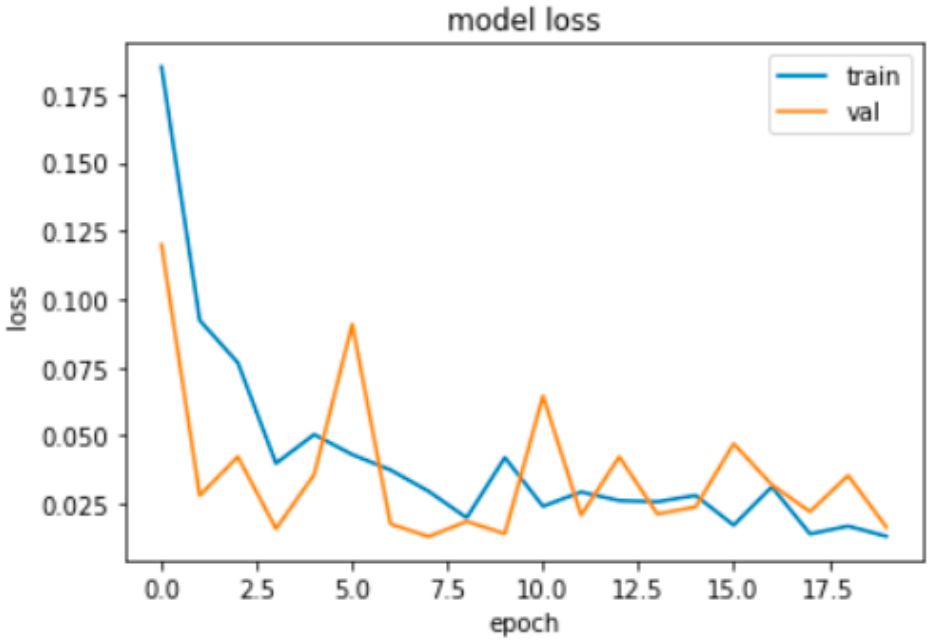


Figure 4. Training and Validation Loss of Image Classifier Model.

The 18 sample images in Figure 5 were intentionally resized to fit A-HiREV's op-tical camera resolution (3376 × 2704 pixels) before going through the classifier model. As the result shows, it has successfully predicted 17 out of the 18 sample test images accurately and wrongly classified the second image as cloudy while it is a clear image. Due to the nature of this image, the model understands that the image has a similarity to images in the cloudy class more than the ones in the clear class. Table 2 shows a summary of the classifier's prediction performance in classifying images in the entire test set. As the confusion matrix shows, out

of the 3200 test images, only six clear im-ages are classified as cloudy while the classifier has successfully identified all the 1600 cloudy images into the cloudy class.

Table 2. Confusion Matrix Summary of Prediction Performance.

		Predicted by the Classifier	
		Clear	**Cloudy**
Actual Classes	Clear	1594	6
	Cloudy	0	1600

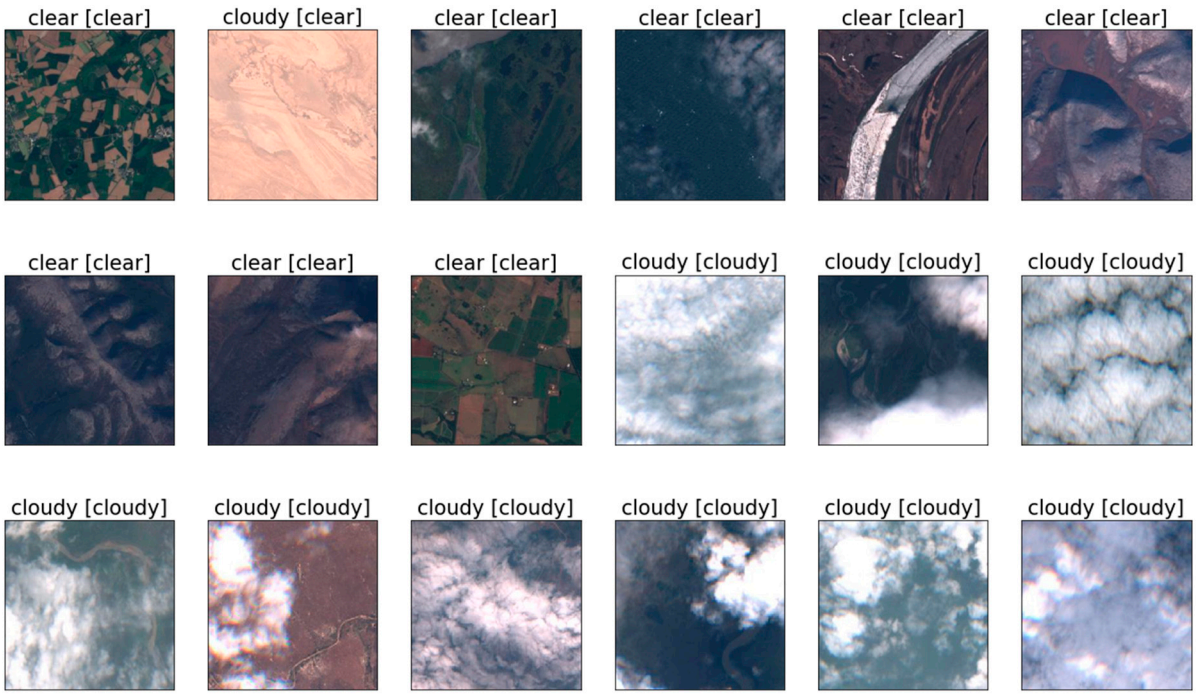


Figure 5. Predicted and Actual Classes of Sample Images in the Test Set.

Freezing the trained model and evaluating its performance is required before pruning and quantization processes are applied to the model to make it fit into the Zedboard. The Deep Neural Network Compiler (DNNC) is a tool in DNNDK that compiles the resulting model and creates a file executable by the Zedboard during inference.

For evaluating the inference performance of the quantized model, we tested the model with 100 images from each class and it showed a 92% overall classification accuracy. The reduction in classification performance is attributed to the pruning and quantization of the model.

5.2. Autonomous Task Scheduling

5.2.1. MDP Formulation of Task Scheduling

The satellite task scheduling problem can be formulated as an MDP problem. The satellite is considered as an agent being in a certain state. The execution of imaging, downlinking, and other commands will change its state, and based on the ultimate objective defied by the ground operators, the satellite's decision to execute such commands and transit to another state will be responded to by rewards computed based on the reward function.

In this study, we considered the A-HiREV satellite as a member of the assumed CubeSat constellation. A-HiREV is flying at an altitude of 400 KM with a 51.6-degree inclination. We designed two ground control stations, one in South Korea and one in Ethiopia. The image acquisition tasks are targeting 27 different cities located in Ethiopia

and South Korea. Figure 6 shows target areas and ground control stations as simulated in Systems Tool Kit (STK). The detail of representing our scenario in MDP formulation needs to define the problem in the form of states, actions, transition function, and rewards.

State Space: Six different types of tasks (states) are considered while designing the autonomous satellite task scheduling algorithm. The task types are Onboard Image Classification (C), Downlinking (D), Ground Station Pointing (G), Imaging (I), Nadir Pointing (N), and Sun Pointing (S).

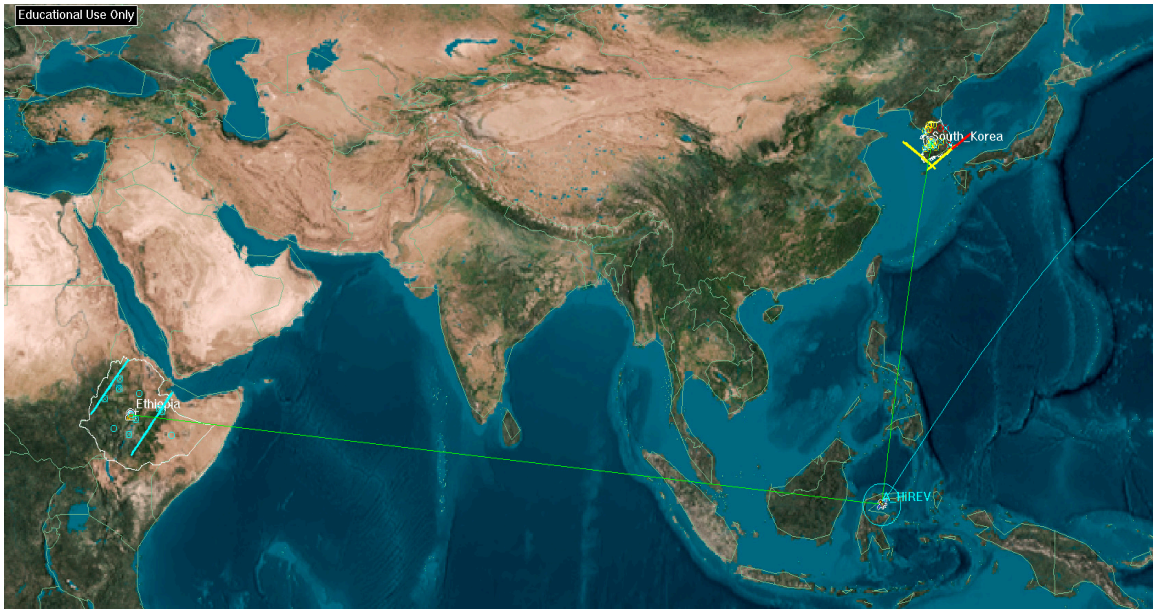


Figure 6. STK Simulation of Target Areas and Ground Stations.

We used the J4 Perturbation propagator while simulating the propagator in STK for a two day scenario analysis period. We used a true-of-date coordinate system and classical coordinate type that has the satellite's orbital elements (semi-major axis: 6778 km, eccentricity: 3.58832×10^{-16}, inclination:51.6 deg, an argument of perigee: 0 deg, RAAN:359.992 deg). The orbit epoch time starts on 20 May 2020 at 00:00:00.000 UTCG and with a step size of 60 s.

When imaging tasks are planned for a given duration (in seconds) to take an image of a specific location (latitude, longitude), the satellite's attitude propagator will be referenced to obtain the time at which the satellite will reach the specified location. The same is true when downlinking tasks are planned to downlink acquired images down to the ground stations; the contact time at which the satellite will be in contact with the ground stations (specified with its latitude and longitude) will be referenced from the propagator.

For imaging and downlinking tasks, ground operators provide starting and ending times of image acquisition or ground contact times based on the propagator. The other tasks, ground station pointing, nadir pointing, sun pointing, and image classification tasks are to be automatically added by the scheduler. Sun-pointing tasks are generated by avoiding the eclipse times of the satellite as generated from the STK simulation.

Accordingly, Tables 3 and 4, respectively, show the list of operator-provided Imaging and Downlinking tasks with their starting and ending times in the assumed scenario. The times are expressed in seconds counting from STK simulation time or mission elapsed time for simplicity.

In Table 3, the notation ('I',6237,6251) represents an Imaging task starting at 6237 s and ending at 6251 s and in Table 4, the notation ('D',347,715) represents a Downlinking task starting at 347 s and ending at 715 s. The access column in Table 4 shows the place of the target ground control station to be contacted.

The operator-supplied imaging and downlinking tasks along with the automatically generated pointing and classification tasks will form the state space in MDP formulation.

Action space: operation tasks are considered as states of the satellite. From a given state, for example, when an Imaging state is completed, the satellite is expected to transit to another possible state (or start performing the next task in time). The action set shown in Table 5 contains the list of possible actions a satellite can perform to transit to the next state. For example, the notation 'I':['G', 'C', 'S', 'I'] in the first row expresses that when the satellite is in an Imaging state, its next possible action is Ground pointing (G), Image Classification (C), Sun pointing (S), or another Imaging (I).

Table 3. List of Imaging Tasks.

Access	Imaging Tasks
1	('I', 6237, 6251)
2	('I', 6241, 6300)
3	('I', 6242, 6289)
4	('I', 6246, 6294)
5	('I', 6251, 6293)
6	('I', 6252, 6285)
7	('I',6253,6321)
8	('I', 6253, 6310)
9	('I', 6286, 6317)
10	('I', 22375, 22416)
11	('I', 22406, 22466)
12	('I', 22417, 22483)
13	('I', 105534, 105583)
14	('I', 105565, 105606)
15	('I', 105588, 105655)
16	('I', 105589, 105656)
17	('I', 118556, 118619)
18	('I', 118562, 118617)
19	('I', 118563, 118574)
20	('I', 118565, 118608)
21	('I', 118566, 118609)
22	('I', 118566, 118614)
23	('I', 118574, 118620)

Transition Function: the transition function maps the current state and its action set into possible next states. For example, if the satellite is in the ground pointing state (G), and if it decides to go to downloading state by taking action 'D', the satellite will transit to downloading state (D).

Reward: the reward (or cost) function is determined by the objective of task scheduling. The environment gives the reward (or cost) to the agent in response to the agent's action. The reward (or cost) of each state can be defined by human experts based on the objective of scheduling. For example, if the satellite is needed to make more image collection or downlinking, more rewards can be given for such tasks. Alternatively, a cost can be attached to tasks based on their onboard memory or power consumption when the satellite is executing them.

In this study, we have considered two rewarding cases. The first one is considering all tasks as equally important tasks with the objective of executing as many tasks as possible in the planning horizon. The second case is weighted importance, in which, tasks are

rewarded based on their consumption or generation of onboard power and usage (or freeing) of onboard memory. We have considered A-HiREV subsystems' profiles while assigning power and memory utilization of tasks and computing rewards of each state. The reward is computed as follows:

$$\mathrm{R(s)} \leftarrow \mathrm{tStart(s)} + (\mathrm{P(s)} + \mathrm{M(s)}) * \mathrm{Conflict(s)} * Priority(s)/Duration(s), \forall \mathrm{s}\, \epsilon\, \mathrm{S} \quad (1)$$

Table 4. List of Downlinking Tasks.

Access	Downlinking Tasks
Kor1	('D', 347, 715)
Kor2	('D', 5955, 6593)
Kor3	('D', 11784, 12360)
Eth1	('D', 16450, 16852)
Kor4	('D', 17705, 18143)
Eth2	('D', 22104, 22730)
Kor5	('D', 23558, 24032)
Kor6	('D', 29326, 29939)
Kor7	('D', 35108, 35717)
Eth3	('D', 57702, 58289)
Eth4	('D', 63500, 64030)
Kor8	('D', 89143, 89752)
Kor9	('D', 94920, 95534)
Kor10	('D', 100827, 101301)
Eth5	('D', 105272, 105903)
Kor11	('D', 106716, 107154)
Eth6	('D', 111177, 111563)
Kor12	('D', 112500, 113075)
Kor13	('D', 118267, 118905)
Kor14	('D', 124145, 124513)
Eth7	('D', 140923, 141396)
Eth8	('D', 146630, 147244)
Kor15	('D', 172355, 172800)

Table 5. Action Set of States.

Current State	Possible Next Action	Notation
Imaging (I)	Ground Pointing (G), Image Classification (C), Sun Pointing (S), Imaging (I)	'I': ['G','C','S', 'I']
Downlinking (D)	Nadir Pointing (N), Sun Pointing (S), Ground Pointing (G)	'D': ['N','S','G']
Sun Pointing (S)	Nadir Pointing (N), Ground Pointing (G), Image Classification(C)	'S': ['N','G', 'C']
Image Classification (C)	Nadir Pointing (N), Ground Pointing (G),Sun Pointing (S)	'C': ['N','G','S']
Nadir Pointing (N)	Imaging (I)	'N': ['I']
Ground Pointing (G)	Downlinking (D)	'G': ['D']

As defined in Equation (1), a reward of the state is computed based on its power consumption (P(s)), memory consumption (M(s)), the number of conflicts with other states (Conflict(s)), its priority, and duration (Duration(s)). To give better chance for tasks in near time than late tasks, we have included the task's starting time (tStart(s)) in computing

the reward. A task with a greater number of conflicting time windows with other tasks has a high cost. When choosing the next task after the initial task, among the alternative candidate tasks, a task with less cost has a better chance to be the next task but the decision is based on its utility value.

Once the RL environment is defined in MDP, we can apply one of the basic RL algorithms. We used the Q-Learning algorithm to generate a sequence of actions the satellite needs to take by evaluating and updating each action's values in every learning cycle. The result is an optimal policy in terms of the sequence of actions the satellite needs to take at each state from the initial up to the end of the planning horizon.

The Q-value of a state is the total amount of reward (or cost) an agent can obtain in the future, starting from that state and following all possible transitions from the state. Q-values indicate the long-term importance of states after considering the states that follow these states.

Rewards are given directly by the environment, but Q-values must be computed and re-computed from the sequences of actions an agent makes over its entire lifetime. An optimal policy is a policy that attains the optimal Q-values for every state.

Q-values are computed by using the Temporal Difference (TD) equation and the well-known Bellman equation.

Temporal Difference (TD): a way of computing how much the Q-value of the action taken in the previous state has to be changed based on what the agent has learned about the Q-values of the current state's actions. It is computed as:

$$\mathrm{TD}(s_t, \mathrm{a}_t) = r_t + \gamma . \min_{\mathrm{a} \in \mathrm{A}} Q(s_{t+1}, \mathrm{a}) - \mathrm{Q}(s_t, \mathrm{a}_t) \tag{2}$$

$\mathrm{TD}(s_t, \mathrm{a}_t)$ is temporal difference for the action taken in the previous state,
r_t is the reward received for the action taken in the previous state,
γ is the discount factor (between 0 and 1),
$\min_{\mathrm{a} \in \mathrm{A}} Q(s_{t+1}, \mathrm{a})$ is the least Q-value available for any action in the current state,
$\mathrm{Q}(s_t, \mathrm{a}_t)$ is the Q-value for the action taken in the previous state.

In computing Q-values, we first initialize the Q-table of all actions in states by zero and continuously update the Q-value of each state's actions using the temporal difference value we obtain as the agent transits to the next states. So, the new Q-value of every action in a state is computed as:

$$\mathrm{Q}^{new}(s_t, \mathrm{a}_t) = \mathrm{Q}^{old}(s_t, \mathrm{a}_t) + \alpha . \mathrm{TD}(s_t, \mathrm{a}_t) \tag{3}$$

$\mathrm{Q}^{new}(s_t, \mathrm{a}_t)$ is the new Q-value for the action taken in the previous state,
$\mathrm{Q}^{old}(s_t, \mathrm{a}_t)$ is the old Q-value for the action taken in the previous state,
α is the learning rate (between 0 and 1),
$\mathrm{TD}(s_t, \mathrm{a}_t)$ is the temporal difference for the action taken in the previous state.

When there is consideration of onboard resources such as onboard memory and power, states will obtain a high cost for their addition into onboard memory or consumption of onboard power whereas they receive less cost value for freeing memory and generating additional power. Finally, the states (tasks) with less value will be preferred over the other alternative tasks to transit from a particular state.

5.2.2. Scheduling Algorithm

The Q-Learning-based scheduling algorithm starts by formulating the onboard satellite scheduling problem in a way suitable for MDP and when executed, it returns the task execution plan (schedule) as a final output. We implemented the algorithm in python and evaluated its outcome. Figure 7 shows a flow chart of the scheduling algorithm.

In the inference mode, given a starting state, the scheduler returns a sequence of the next states leading up to the last state in the planning horizon. While going towards the final state, it keeps on collecting rewards by following imaging, downlinking, and

classification tasks. In doing so, the machine is achieving the goal of the scheduling, which is to plan as many imaging, classification, and downlinking tasks as possible.

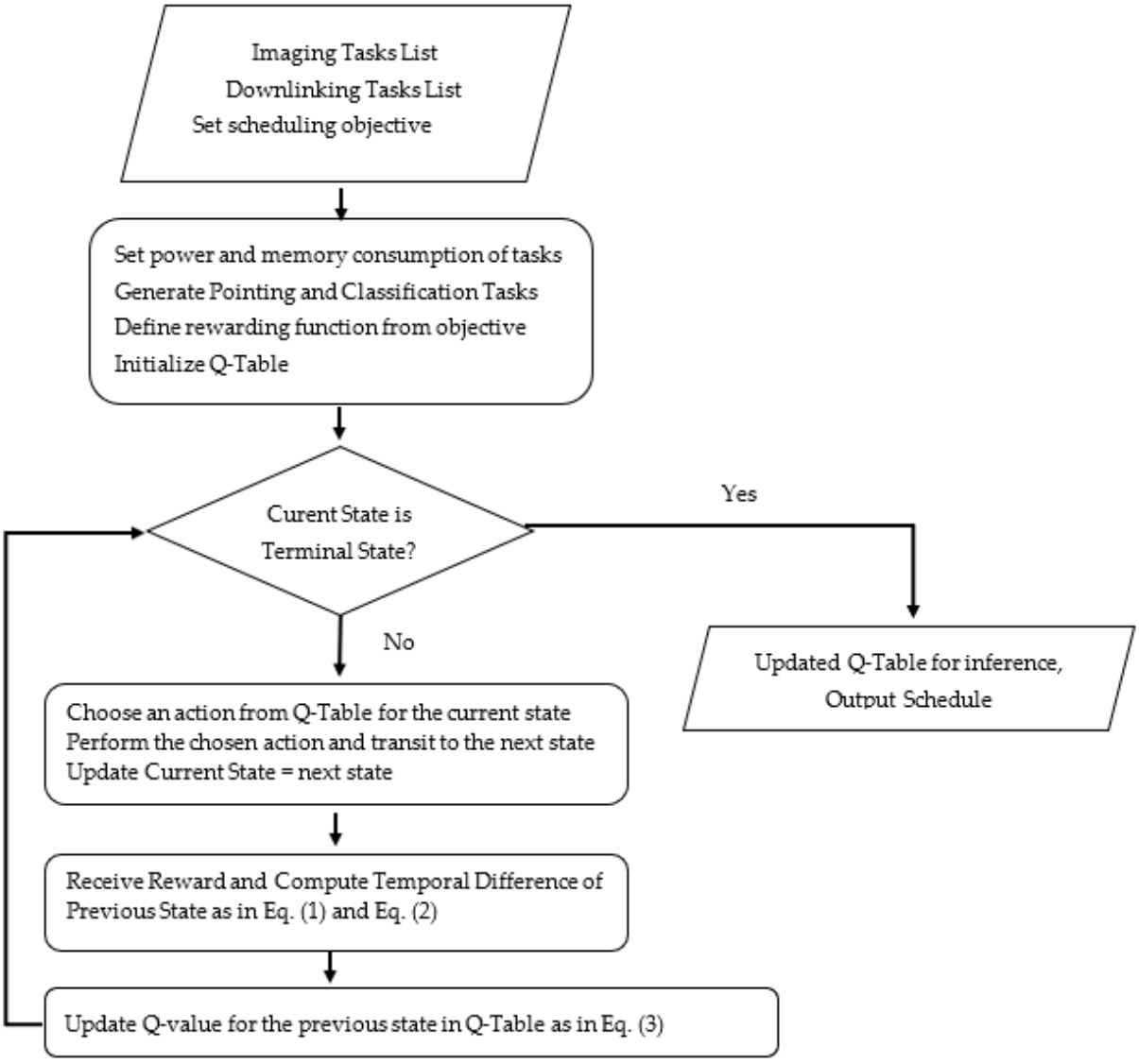


Figure 7. Q-Learning Based Scheduling Flow Chart.

5.2.3. Simulation Results and Discussion

We simulated the algorithm in two cases by altering the objective of scheduling. The first case considers all tasks as equally important and the second case, weighted importance, considers less costly tasks as important. The resulting schedule in the first case deals with the scheduling of tasks based on the order of their starting times as long as the scheduling horizon duration. The resulting schedule in this case does not guarantee the effective utilization of onboard resources.

In the second, weighted importance case, tasks are evaluated based on their onboard resource usage and generation. Additionally, in this case, we evaluated the need of including onboard image classification as one of the tasks in the scheduling. The simulation result shows that the RL-based algorithm autonomously includes important onboard image classification tasks at selected times as they clear out onboard memory and utilize less power. Table 6 shows the resulting list of tasks chosen by the scheduling algorithm. Starting from the initial task in time, the algorithm has evaluated the best next tasks based on the Q-values of each action in tasks. Hence, the RL-based onboard scheduling algorithm

enables the satellite to autonomously make optimal schedules considering its onboard resources. When a need arises from ground operators to execute high-priority tasks or the inclusion of new tasks into the schedule, the algorithm can reschedule the remaining tasks in time after every contact with the ground stations.

Table 6. RL-Based Scheduling Result.

Order	Scheduled Task
1	(‘G’, 316, 346)
2	(‘D’, 347, 715)
3	(‘S’, 716, 6205)
4	(‘N’, 6206, 6236)
5	(‘I’, 6237, 6251)
6	(‘I’, 6252, 6285)
7	(‘I’, 6286, 6317)
8	(‘C’, 6318, 6348)
9	(‘S’, 6349, 22343)
10	(‘N’, 22344, 22374)
11	(‘I’, 22375, 22416)
12	(‘I’, 22417, 22483)
13	(‘C’, 22484, 22514)
14	(‘S’, 22515, 105502)
15	(‘N’, 105503, 105533)
16	(‘I’, 105534, 105583)
17	(‘I’, 105588, 105655)
18	(‘C’, 105656, 105686)
19	(‘S’, 105687, 118524)
20	(‘N’, 118525, 118555)
21	(‘I’, 118556, 118619)
22	(‘C’, 118620, 118650)
23	(‘S’, 118651, 124113)
24	(‘G’, 124114, 124144)
25	(‘D’, 124145, 124513)
26	(‘S’, 124514, 140891)
27	(‘G’, 140892, 140922)
28	(‘D’, 140923, 141396)
29	(‘S’, 141397, 146598)
30	(‘G’, 146599, 146629)
31	(‘D’, 146630, 147244)
32	(‘S’, 149113, 172323)
33	(‘G’, 172324, 172354)
34	(‘D’, 172355, 172800)

In this study, we found that the satellite’s total contact time duration with the two ground stations is much less than the image acquisition times. In addition, the satellite’s high-resolution image acquisition camera generates data at a rate higher than the download

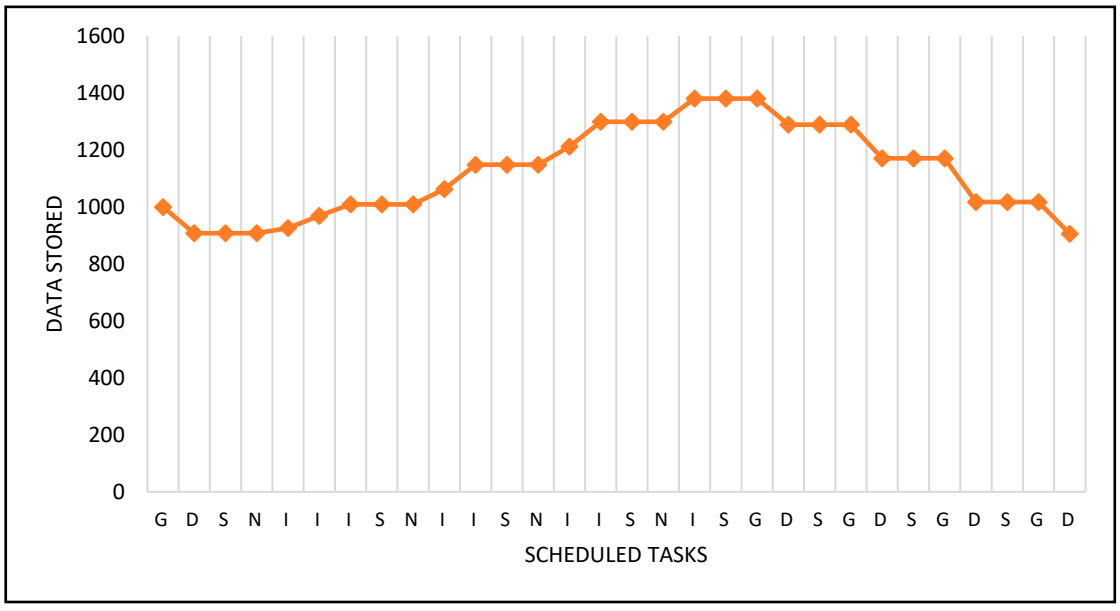
bandwidth of its S-band antenna. This results in the satellite's onboard memory becoming full of acquired images which are mostly cloudy and useless for Earth observation missions.

Figure 8a,b show onboard memory utilization of tasks scheduled in the two-days planning horizon. We assumed an instance of onboard memory with 1 GB of image data as an initial point. Figure 8a shows the memory utilization of scheduled tasks in the absence of onboard image classification, whereas Figure 8b shows the memory utilization when onboard image classification is included as a task. As clearly observed from Figure 8a, the imaging tasks keep on adding more data into memory and the rate of downlinking speed could not cope with this increase in data storage in the absence of onboard image classification tasks. Whereas, in Figure 8b, the onboard image classification has dramatically reduced onboard data as cloudy images are cleared out without wasting scarce onboard memory. This result strongly suggests the need for onboard image processing tasks to clear out cloudy images before they are downlinked to ground stations. This way, we can save onboard memory as well as the power and bandwidth needed for downlinking. This in turn results in the effective utilization of satellite contact times with ground stations for quick transmission of clear Earth observation images.

Finally, we have made a heuristic-based schedule aimed at selecting the next tasks with minimum resource requirements to minimize the overall resource consumption during the planning horizon. In this case, tasks next to the current task are evaluated with their onboard resource demand and the task with a minimum request will be chosen.

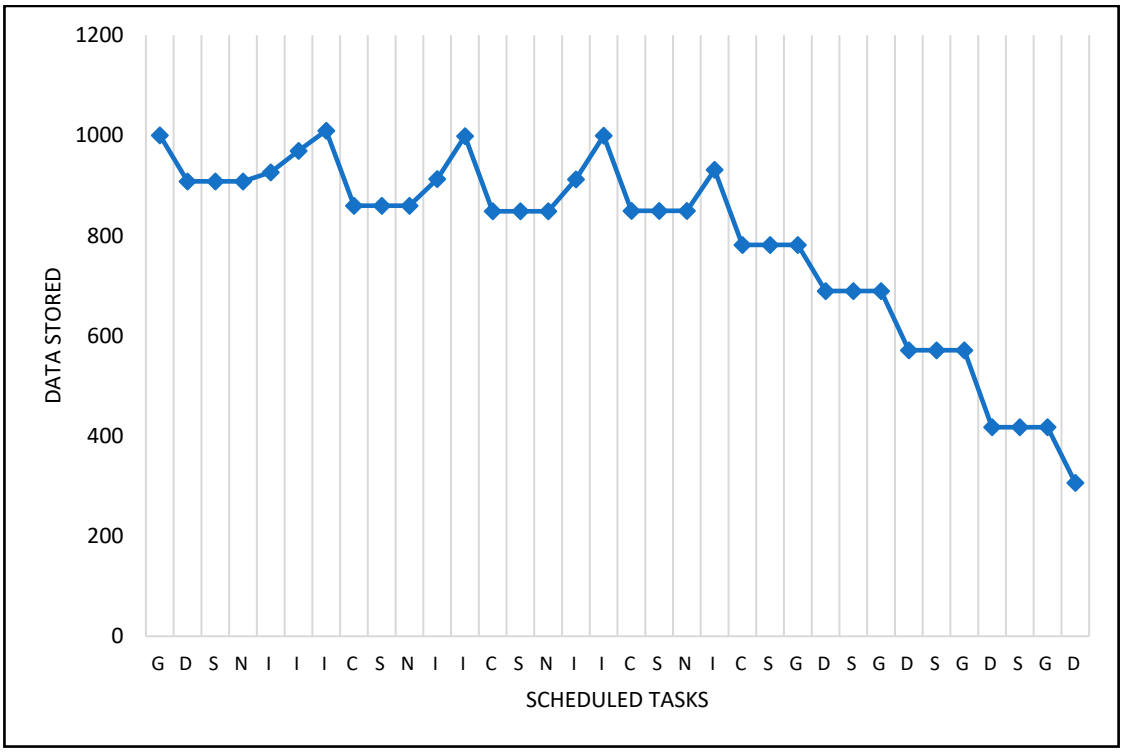
To illustrate this case, we chose an instance imaging task, ('I', 6237, 6251) and considered it as the satellite's current state. As shown in Table 5 (Action set of states), the satellite can take one of the actions from imaging, classification, sun-pointing, and ground-pointing to transit into its next state. Figure 9 illustrates this scenario with the alternative next states' values and costs as evaluated by the Q-value and cost functions, respectively. In the case of the heuristic method, the satellite's next task will be the task with the least cost,('C', 6252, 6282) whereas in the case of the RL-based method, the task with the least value, ('I', 6252, 6285), will be scheduled as the next task.

The heuristic-based scheduling algorithm executes quite quickly compared to RL-based scheduling. However, as it greedily searches for immediate next tasks with minimum requirement, it fails to consider overall optimization on the horizon and missed scheduling important imaging tasks which could have been accompanied by upcoming image classification tasks. Table 7 shows this result of heuristic-based scheduling.



(**a**)

Figure 8. *Cont.*



(**b**)

Figure 8. (**a**) Onboard Memory Utilization without Classification; (**b**) Onboard Memory Utilization with Classification.

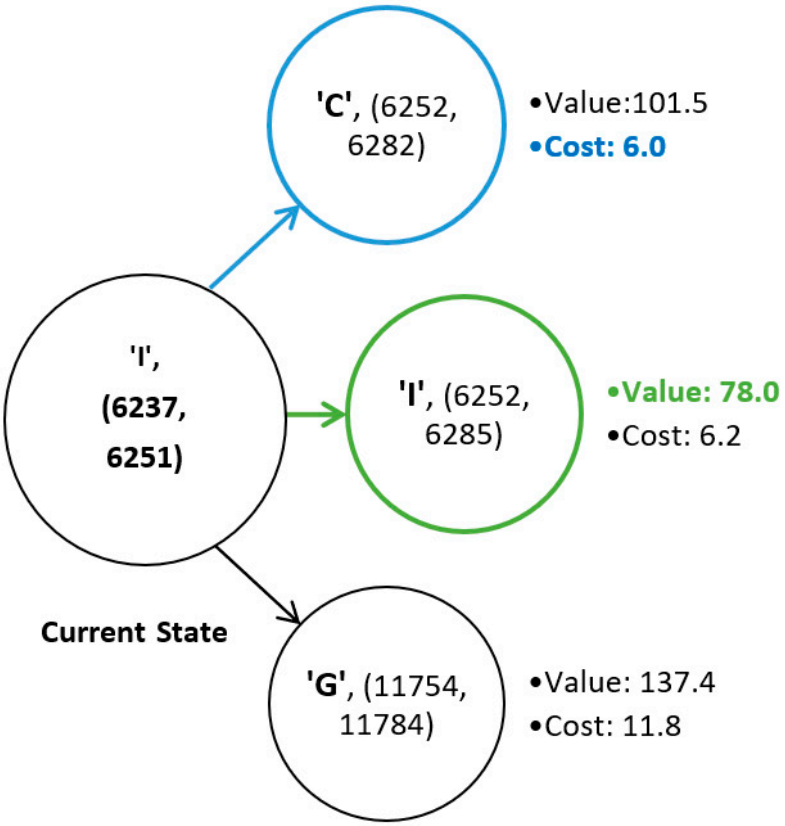


Figure 9. Sample Value and Cost of Next States.

Table 7. Heuristic-Based Scheduling Result.

Order	Scheduled Task
1	(‘G’, 316, 346)
2	(‘D’, 347, 715)
3	(‘S’, 716, 6205)
4	(‘N’, 6206, 6236)
5	(‘I’, 6237, 6251)
6	(‘C’, 6252, 6282)
7	(‘S’, 6283, 11752)
8	(‘G’, 11753, 11783)
9	(‘D’, 11784, 12360)
10	(‘S’, 12361, 22343)
11	(‘N’, 22344, 22374)
12	(‘I’, 22375, 22416)
13	(‘C’, 22417, 22447)
14	(‘G’, 23527, 23557)
15	(‘D’, 23558, 24032)
16	(‘S’, 24033, 105502)
17	(‘N’, 105503, 105533)
18	(‘I’, 105534, 105583)
19	(‘C’, 105584, 105614)
20	(‘G’, 106685, 106715)
21	(‘D’, 106716, 107154)
22	(‘S’, 107155, 118524)
23	(‘N’, 118525, 118555)
24	(‘I’, 118556, 118619)
25	(‘C’, 118620, 118650)
26	(‘S’, 118651, 124113)
27	(‘G’, 124114, 124144)
28	(‘D’, 124145, 124513)
29	(‘S’, 124514, 124573)

Out of the 23 operator-provided imaging tasks, indicated in Table 3, only eight are feasible due to conflict in the observation window of target points. The resulting schedules of both RL-based and heuristic-based scheduling, Tables 6 and 7 respectively, show how many of these imaging tasks are scheduled. In the heuristic case, only four of these tasks are scheduled, which is 50% of the possible imaging tasks. Whereas in the RL case, seven tasks are included in the schedule, which is more than 85% of the feasible imaging tasks. This ensures the RL-based scheduling is fitting in line with the ultimate objective of planning as many imaging and downlinking tasks while ensuring efficient resource utilization. This situation becomes more significant when applied to the constellation of similar CubeSats or even with the addition of more imaging tasks in the planning horizon.

6. Conclusions

The efficiency of satellite constellations depends on the efficiency of individual satellites in the constellation. This study assumes the development of a constellation of CubeSats

of standard performance with general memory and focused on ensuring efficient resource utilization by experimenting on one of the CubeSats in the constellation.

For this purpose, we have developed and tested a custom convolutional neural network-based onboard image classification model which fits into the processor of the 6-unit CubeSat (A-HiREV). The model is trained and tested with a freely available Sentinel-2 dataset. The onboard image classifier classified images stored in onboard memory into cloudy and clear image classes effectively. With the help of this classifier, the satellite can solve the problem of flooding onboard memory with cloudy images and also avoid wastage of bandwidth, power, and time in transmitting these images to ground stations.

The second part of this paper focused on designing an RL-based onboard autonomous task scheduler that is utilized by the satellite considering its resources in real-time. This proposed method is tested to make effective schedules of imaging, downlinking, pointing, and image classification tasks. The algorithm iterates from the initial task by evaluating all tasks and picking the best next tasks not only based on their onboard resource requirement but also their contribution to making an overall optimal schedule. The ultimate objective of the scheduling is planning as many imaging and downlinking tasks as possible while ensuring efficient resource utilization. Compared to a heuristic-based scheduling algorithm designed for the same objective, RL-based scheduling has performed better in scheduling more imaging and downlinking tasks while ensuring efficient resource utilization. In the RL-based scheduling, more than 85% of the feasible imaging tasks are scheduled while only 50% are scheduled in the heuristic-based scheduling. Both methods have scheduled all of the feasible downlinking tasks. As more imaging tasks are scheduled in RL-based scheduling, more onboard memory is required. However, the properly scheduled onboard image classification tasks have enabled the system to effectively utilize onboard memory by identifying and discarding cloudy images.

Hence, the two parts of this study jointly ensured efficient utilization of onboard memory, power, and bandwidth in the highly resource-constrained CubeSat platforms and their constellations. The study also highlighted the importance of granting controlled autonomy to satellites as seen in both onboard image classification and onboard task scheduling techniques. As a result, more than 85% of feasible imaging tasks and all feasible downlinking tasks are scheduled in the two-day's planning horizon. The properly scheduled onboard image classification tasks also protected the onboard memory from flooding with useless cloudy images. Therefore, we recommend the use of RL-based onboard task scheduling including onboard image classification in ordinary CubeSats and their constellations to boost overall mission effectiveness.

In future studies, we will investigate the addition of inter-satellite communication tasks in onboard scheduling and their effect on efficient resource utilization. The result of this research can be used as input to further studies regarding constellation-level resource utilization.

Author Contributions: Conceptualization, D.A.Z.; methodology, D.A.Z.; software, D.A.Z.; validation, D.A.Z. and H.-D.K.; formal analysis, D.A.Z.; investigation, D.A.Z.; resources, D.A.Z. and H.-D.K.; data curation, D.A.Z.; writing—original draft preparation, D.A.Z.; writing—review and editing, H.-D.K.; visualization, D.A.Z.; supervision, H.-D.K.; project administration, H.-D.K.; funding acquisition, H.-D.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea funded by the Ministry of Science and ICT under Grant (2022M1A3C2074536, Future Space Education Center) and also was supported by the research grant of the Gyeongsang National University in 2022.

Data Availability Statement: We used an openly released dataset of Sentinel-2 satellite images used in [32] and available in Harvard dataverse website.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Giuffrida, G.; Diana, L.; De Gioia, F.; Benelli, G.; Meoni, G.; Donati, M.; Fanucci, L. CloudScout: A Deep Neural Network for On-Board Cloud Detection on Hyperspectral Images. *Remote. Sens.* **2020**, *12*, 2205. [CrossRef]
2. Feruglio, L.; Corpino, S. Neural networks to increase the autonomy of interplanetary nanosatellite missions. *Robot. Auton. Syst.* **2017**, *93*, 52–60. [CrossRef]
3. Chien, S.; Doubleday, J.; Thompson, D.R.; Wagstaff, K.L.; Bellardo, J.; Francis, C.; Baumgarten, E.; Williams, A.; Yee, E.; Stanton, E.; et al. Onboard Autonomy on the Intelligent Payload EXperiment CubeSat Mission. *J. Aerosp. Inf. Syst.* **2016**, *14*, 307–315. [CrossRef]
4. Arechiga, A.P.; Michaels, A.J.; Black, J.T. Onboard Image Processing for Small Satellites. In Proceedings of the NAECON 2018—IEEE National Aerospace and Electronics Conference, Dayton, OH, USA, 23–26 July 2018; pp. 234–240.
5. Fraire, J.A.; Nies, G.; Gerstacker, C.; Hermanns, H.; Bay, K.; Bisgaard, M. Battery-Aware Contact Plan Design for LEO Satellite Constellations: The Ulloriaq Case Study. *IEEE Trans. Green Commun. Netw.* **2020**, *4*, 236–245. [CrossRef]
6. Azami, M.; Orger, N.C.; Schulz, V.H.; Oshiro, T.; Chio, M. Earth Observation Mission of a 6U CubeSat with a 5-Meter Resolution for Wildfire Image Classification Using Convolution Neural Network Approach. *Remote Sens.* **2022**, *14*, 1874. [CrossRef]
7. Curzi, G.; Modenini, D.; Tortora, P. Large Constellations of Small Satellites: A Survey of Near Future Challenges and Missions. *Aerospace* **2020**, *7*, 133. [CrossRef]
8. Liang, J.; Zhu, Y.-H.; Luo, Y.-Z.; Zhang, J.-C.; Zhu, H. A precedence-rule-based heuristic for satellite onboard activity planning. *Acta Astronaut.* **2021**, *178*, 757–772. [CrossRef]
9. Mok, S.-H.; Jo, S.; Bang, H.; Leeghim, H. Heuristic-Based Mission Planning for an Agile Earth Observation Satellite. *Int. J. Aeronaut. Space Sci.* **2019**, *20*, 781–791. [CrossRef]
10. Jo, S.P.; Jaehwan, P.; Bang, H. Mission Scheduling for SAR Satellite Constellations with a Heuristic Approach. In Proceedings of the 30th International Symposium on Space Technology and Science, Kobe-Hyogo, Japan, 4–10 July 2015.
11. Peng, S.; Chen, H.; Du, C.; Li, J.; Jing, N. Onboard Observation Task Planning for an Autonomous Earth Observation Satellite Using Long Short-Term Memory. *IEEE Access* **2018**, *6*, 65118–65129. [CrossRef]
12. Li, Z.; Xu, R.; Cui, P.; Zhu, S. Artificial Neural Network Based Mission Planning Mechanism for Spacecraft. *Int. J. Aeronaut. Space Sci.* **2018**, *19*, 111–119. [CrossRef]
13. Li, G.; Xing, L.; Chen, Y. A hybrid online scheduling mechanism with revision and progressive techniques for autonomous Earth observation satellite. *Acta Astronaut.* **2017**, *140*, 308–321. [CrossRef]
14. Yao, F.; Li, J.; Chen, Y.; Chu, X.; Zhao, B. Task allocation strategies for cooperative task planning of multi-autonomous satellite constellation. *Adv. Space Res.* **2019**, *63*, 1073–1084. [CrossRef]
15. Miralles, P.; Scannapieco, A.F.; Jagadam, N.; Baranwal, P.; Faldu, B.; Abhang, R.; Bhatia, S.; Bonnart, S.; Bhatnagar, I.; Prasad, P.; et al. Machine Learning in Earth Observation Operations: A review. In Proceedings of the 72nd International Astronautical Congress (IAC), Dubai, United Arab Emirates, 25–29 October 2021.
16. Cho, D.-H.; Choi, W.-S.; Kim, M.-K.; Kim, J.-H.; Sim, E.; Kim, H.-D. High-Resolution Image and Video CubeSat (HiREV): Development of Space Technology Test Platform Using a Low-Cost CubeSat Platform. *Int. J. Aerosp. Eng.* **2019**, *2019*, 8916416. [CrossRef]
17. Zhang, Z.; Iwasaki, A.; Xu, G.; Song, J. Cloud detection on small satellites based on lightweight U-net and image compression. *J. Appl. Remote. Sens.* **2019**, *13*, 026502. [CrossRef]
18. Hughes, M.J.; Hayes, D. Automated Detection of Cloud and Cloud Shadow in Single-Date Landsat Imagery Using Neural Networks and Spatial Post-Processing. *Remote. Sens.* **2014**, *6*, 4907–4926. [CrossRef]
19. King, M.D.; Platnick, S.; Menzel, W.P.; Ackerman, S.A.; Hubanks, P.A. Spatial and Temporal Distribution of Clouds Observed by MODIS Onboard the Terra and Aqua Satellites. *IEEE Trans. Geosci. Remote. Sens.* **2013**, *51*, 3826–3852. [CrossRef]
20. Ma, L.; Liu, Y.; Zhang, X.; Ye, Y.; Yin, G.; Johnson, B.A. Deep learning in remote sensing applications: A meta-analysis and review. *ISPRS J. Photogramm. Remote Sens.* **2019**, *152*, 166–177. [CrossRef]
21. Flamis, G.; Kalapothas, S.; Kitsos, P. Workflow on CNN utilization and inference in FPGA for embedded applications. In Proceedings of the 6th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM 2021), Preveza, Greece, 24–26 September 2021; pp. 1–6.
22. Chen, J.; Ran, X. Deep Learning with Edge Computing: A Review. *Proc. IEEE* **2019**, *107*, 1655–1674. [CrossRef]
23. Vestias, M.P.; Duarte, R.P.; De Sousa, J.T.; Neto, H.C. A Configurable Architecture for Running Hybrid Convolutional Neural Networks in Low-Density FPGAs. *IEEE Access* **2020**, *8*, 107229–107243. [CrossRef]
24. XILINK. *DNNDK User Guide v1.6. 2019*; XILINK: San Jose, CA, USA, 2019.
25. Lee, J.; Kim, H.; Chung, H.; Ko, K. Genetic algorithm-based scheduling for ground support of multiple satellites and antennae considering operation modes. *Int. J. Aeronaut. Space Sci.* **2016**, *17*, 89–100. [CrossRef]
26. Zhang, J.; Xing, L. An improved genetic algorithm for the integrated satellite imaging and data transmission scheduling problem. *Comput. Oper. Res.* **2022**, *139*, 105626. [CrossRef]
27. Chen, X.; Reinelt, G.; Dai, G.; Spitz, A. A mixed integer linear programming model for multi-satellite scheduling. *Eur. J. Oper. Res.* **2019**, *275*, 694–707. [CrossRef]

28. Qu, Q.; Liu, K.; Li, X.; Zhou, Y.; Lu, J. Satellite Observation and Data-Transmission Scheduling using Imitation Learning based on Mixed Integer Linear Programming. In *IEEE Transactions on Aerospace and Electronic Systems*; IEEE: New York, NY, USA, 2022; pp. 1–25.
29. Kim, J.; Ahn, J.; Choi, H.-L.; Cho, D.-H. Task Scheduling of Agile Satellites with Transition Time and Stereoscopic Imaging Constraints. *J. Aerosp. Inf. Syst.* **2020**, *17*, 285–293. [CrossRef]
30. Huang, Y.; Mu, Z.; Wu, S.; Cui, B.; Duan, Y. Revising the Observation Satellite Scheduling Problem Based on Deep Reinforcement Learning. *Remote Sens.* **2021**, *13*, 2377. [CrossRef]
31. Wen, Z.; Li, L.; Song, J.; Zhang, S.; Hu, H. Scheduling single-satellite observation and transmission tasks by using hybrid Actor-Critic reinforcement learning. *Adv. Space Res.* 2022; *in press*. [CrossRef]
32. Sarukkai, V.; Jain, A.; Uzkent, B.; Ermon, S. Cloud Removal in Satellite Images Using Spatiotemporal Generative Networks. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), Snowmass, CO, USA, 1–5 March 2020; pp. 1785–1794.