MULTI-MODAL GLOBAL SURVEILLANCE METHODOLOGY FOR PREDICTIVE AND ON-DEMAND CHARACTERIZATION OF LOCALIZED PROCESSES USING CUBE SATELLITE PLATFORMS

A Thesis

by

MARIO MONDELLINI MENDOZA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Pavel V. Tsvetkov
Committee Members,	Sunil S. Chirayath
	Michael B. Pate
Head of Department,	Michael Nastasi

May 2021

Major Subject: Nuclear Engineering

ABSTRACT

Cube satellites (CubeSats) present a unique platform for monitoring localized processes anywhere within the Earth's surface or atmosphere using novel data analysis techniques. Areas of interest can be targeted at certain times on an on-demand basis by storing the CubeSat constellation onboard the International Space Station (ISS). CubeSats equipped with adequate sensors and data analytics capabilities can create an autonomous characterization surveillance method for the phenomena of interest. CubeSats are advantageous over conventional satellites for remote monitoring because of their reduced costs and higher simplicity due to the availability of commercially-off-the-shelf components. The work presented in this thesis contributed to the eventual deployment of the CubeSat surveillance system by laying down a basis for the overall methodology.

The CubeSat surveillance system focuses on phenomena of immediate interest divided into three categories surrounding the nuclear fuel cycle; vehicles, facilities and infrastructural emergencies, and construction/mining events of interest. To observe the phenomena, a constellation of 3U and 6U CubeSats deployed from the ISS with adequate components was chosen. The constellation achieves inter-satellite communications through additional satellite network relays and ground communications through a network of ground stations. To adequately observe the phenomena, four different sensor configurations were identified: panchromatic/multispectral in the visible and near-infrared spectrum, multispectral in infrared spectrum, hyperspectral in infrared spectrum, and multispectral in ultraviolet spectrum. While a panchromatic/multispectral sensor

configuration has CubeSat flight heritage at the required spatial resolutions, the other three sensor types need future development to meet signature and system requirements. Once each sensor onboard the CubeSat system collects data on a target of interest, the onboard computers apply the machine learning based characterization methodology to identify phenomena. Four surrogate datasets containing representative simplified “images” were created for each sensor type to train the characterization methodology. A convolutional neural network was applied to each dataset and they produced recall rates for the phenomena between 89.7% - 99.3% and precision rates between 92.3% - 99.9%. Each phenomenon’s presence probability from each network is then combined into a final characterization solution for a target area. The thesis also discusses the applications of the CubeSat surveillance methodology for microreactor deployment.

DEDICATION

Este trabajo se lo dedico a mi familia, mis amigos y compañeros, los días trabajando sin fin, y a los litros de cafeína que se consumieron.

I dedicate this work to my family, my friends and colleagues, the long days of work without end, and to the gallons of caffeine which were consumed.

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Tsvetkov, and my committee members, Dr. Chirayath and Dr. Pate, for their guidance and support throughout the course of this research.

Thanks to Mike Lewis and Troy Guy from NanoRacks for their contributions and guidance on the satellite aspects of the thesis.

To David Peters, John Dickinson, and John van der Laan from Sandia National Lab for their guidance and input on various aspects of the project.

To Alan Aguilar and Ulises Eduardo Garzon Nunez for their help on figuring out satellite orbital modelling.

Thanks also go to my friends and colleagues and the department faculty and staff for making my time at Texas A&M University a great experience.

Finally, thanks to my family and close friends for their endless encouragement and support.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a thesis committee consisting of Professors Pavel V. Tsvetkov and Sunil S. Chirayath of the Department of Nuclear Engineering and Professor Michael B. Pate of the Department of Mechanical Engineering.

All work conducted for the thesis was completed by the student independently.

Funding Sources

This material is based upon work supported by the Department of Energy/National Nuclear Security Administration under Award Number(s) DE-NA0003921 through the Consortium of Enabling Technologies and Innovation. **Disclaimer:** This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

1. INTRODUCTION

This thesis presents the work done towards the development of a multi-modal global surveillance methodology for predictive and on-demand characterization of localized processes using cube satellite (CubeSat) platforms. This surveillance system onboard miniature satellites called CubeSats presents a highly mobile monitoring platform with access to any point on the globe at times of interest [1]. Equipping CubeSats with appropriate sensors and data analytics capabilities provides the capability of characterizing phenomena on the Earth's surface or atmosphere on an on-demand basis. The novelty of this monitoring system is highlighted in the development of the deep learning techniques for data analytics. CubeSats prove advantageous over other surveillance platforms, such as drones, because they can target areas of interest anywhere in the world in a short amount of time. When compared to larger, conventional surveillance satellites, CubeSats present a much lower cost due to their size, increased simplicity, and use of commercially-of-the-shelf (COTS) components [2,3]. As this novel monitoring methodology is developed under the Department of Nuclear Engineering at Texas A&M University, the functionality of the CubeSat system is applied to a nuclear surveillance use example. This paper does not aim to reinvent any currently existing satellite observation methods for nuclear safeguards verification.

Currently, organizations like the International Atomic Energy Agency (IAEA) and the U.S. Department of Energy National Nuclear Security Administration (DOE/NNSA) along with various space agencies and commercial companies use satellite imagery for safeguards verification as made possible under the Additional Protocol (INFCIRC/540) of

the Non-Proliferation Treaty (NPT) [4,5]. Approved in 1997, the Additional Protocol increased the previously established safeguards measures and expanded them to all NPT States regardless of their Nuclear Weapon State status [4]. Through this expansion of safeguards measures, the use of commercial satellite imagery was introduced for verifying nuclear materials and facilities are not used for clandestine military activities. Satellite imagery can be used to aid in planning of future inspection activities, to verify information provided by States, to identify undeclared activities, and to observe any changes in activities at facilities along the nuclear fuel cycle [6]. The IAEA's Satellite and Imagery Laboratory employs the use of commercial geostationary satellites thousands of kilometers away from the surface of the Earth to collect images of nuclear facilities [7]. Some satellites previously used for image collection include the IKONOS-2 and EROS-AI satellites [7]. In recent years, swarms of small satellites (SmallSats) and CubeSats, from companies such as Planet Labs, have also provided valuable images for the verification of nuclear safeguards [8]. The work accomplished in this thesis is similar to currently used CubeSat systems for nuclear surveillance, but it applies a novel deep learning-based data analytics methodology.

The work in this paper establishes the foundation for a CubeSat surveillance system by developing a base design methodology for the system's capabilities with a focus on its data analytics capabilities. The CubeSat surveillance system features a series of different satellites, each equipped with different sensors capable of distinguishing phenomena of interest. The necessary components and design processes needed for the successful development of a CubeSat are explored and analyzed in this thesis. Different

architecture scenarios were considered for the CubeSat surveillance system. A constellation, or multiple satellite, configuration was chosen, mainly for the increased characterization accuracy that comes from multiple satellites since a single CubeSat can only house one to two types of sensors. Communications between each CubeSat in the system will be achieved through a secondary satellite system further away from the Earth acting as a relay for the transmissions. Through collaboration with the launch provider, NanoRacks LLC, the future CubeSat system will be placed onboard the International Space Station (ISS) for deployment. Storing the satellite constellation onboard the ISS allows for it to reach any point of interest on the globe within two to three days maximum [9]. Once the CubeSat system is deployed, the satellites adopt the same orbit as the ISS for the remainder of their lifetime, allowing them to carry out on-demand surveillance. The signature requirements for detection for each phenomenon the system aims to observe are explored to formulate a final recommendation on the different types of sensors needed for the surveillance system. Once each sensor onboard the CubeSat system collects data on a target of interest, the onboard computers will apply the machine learning based characterization methodology developed in this paper before transmitting information to ground stations on the surface. Having these data analytic capabilities within the architecture of the satellites allows for quicker characterization of potentially time-sensitive phenomena and alleviates strains on data transmissions to ground stations. The characterization methodology developed in this thesis is trained on simplified surrogate datasets representative of the recommended sensor data types.

1.1. Phenomena of Interest

The first step in developing a surveillance system is identifying what type of phenomena are of interest for observation. Defining the phenomena and their characteristics will determine everything else about the surveillance system, like the type of sensors needed and the physical CubeSat architecture. As mentioned, the use example for the CubeSat surveillance system focuses on general aspects of nuclear surveillance. In other words, observe objects or process near facilities in the nuclear fuel cycle, as seen in Figure 1, that indicate the presence of unapproved activities. After careful analysis of a CubeSat's capabilities once in orbit, as will be discussed in later sections of this thesis, a CubeSat surveyor is better suited for observing phenomena of immediate interest on a short-term periodic or on-demand basis. Once deployed at Low Earth Orbit (LEO), a CubeSat's orbit will decay completely within one to two years, after which the satellite will reenter the atmosphere [10]. Also, due to a lack of resources on board a CubeSat, like propulsion systems, it cannot maintain a perfect orbit throughout its lifetime. Once a CubeSat is launched and adopts its orbit, it will slowly start deforming [10]. A deformed orbit means that a CubeSat will still pass over all locations of interest on the planet, but just at different rates [11,12]. Therefore, precise periodic surveillance on a long-term basis is better suited for the currently used geostationary, large satellites.

Given the CubeSat's orbital restrictions and motivations for the project, phenomena of interest for the surveyor were selected. There are many possible situations and events that occur around the world which indicate suspicious activity near facilities of the nuclear fuel cycle. For example, the presence of increased construction activities at an

enrichment facility which were not approved by regulatory committees, or a terrorist organization steeling a radioactive isotope and transporting it by truck. A variety of possible scenarios translates to a diversity of objects and processes which the CubeSat system must successfully observe. Therefore, the example phenomena of interest chosen for the CubeSat surveillance system developed in this thesis are automobiles, airplanes, nuclear facilities, infrastructural emergencies, construction activities, and mining activities. It is important to mention that these phenomena pertain to the use example of the monitoring system for nuclear surveillance and not a complete list of phenomena of interest for actual safeguards verifications processes. The importance of surveilling these phenomena along with their specific parameters for observation are discussed in the next few subsections. It is important to specify the parameters characterizing the phenomena to facilitate in the autonomy and sensor selection of the CubeSat system. Instead of transmitting all its raw data to a ground station for human processing after each observation, the CubeSat's data analytics capabilities will autonomously conduct the characterization of the retrieved data before sending it to the surface. Defining specific parameters for phenomena beforehand determines the types of sensors needed which then specify the type of data the characterization methodology is trained on. By training on an adequate surrogate data set, the methodology will learn the necessary features from the data given to characterize a phenomenon in future data. Therefore, the parameters listed in the following subsections for the phenomena of interest are used to inform the sensor selection for the surveillance system.

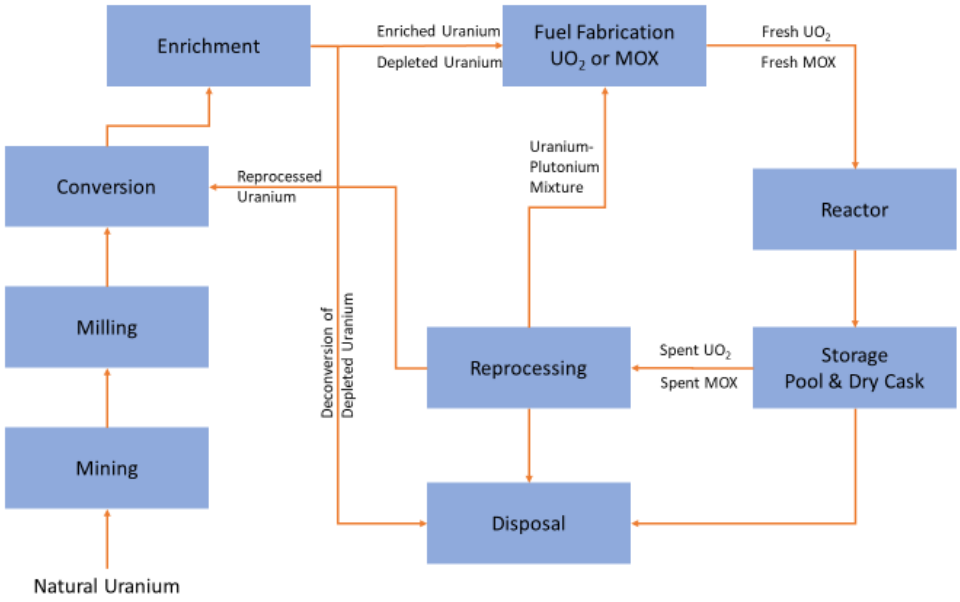


Figure 1. Each step of the nuclear fuel cycle.

1.1.1. Vehicles of Interest

The detection of automobiles can be of great importance while surveilling facilities in the nuclear fuel cycle. An increase in the number of cars at facilities can indicate to the occurrence of suspicious activity, or the surveillance of a vehicle with stolen radioactive material can prove beneficial for the appropriate authorities. There are many parameters that aid in the identification of a car, such as size, speed, temperature, and emissions. For size, there is a variety of different types of vehicles, all with different dimensions. Table 1 and Table 2 separate automobiles into two categories, passenger vehicles and commercial vehicles. For Table 1, the maximum and minimum values are recorded for each dimension. The maximum values reflect those of big “passenger vans”, which have the biggest dimensions out of all passenger vehicles, and the minimum values reflect those of “city cars”, which have the smallest dimensions [13]. These dimensions are for all

available new vehicle models in Europe [13]. For Table 2, the maximum values from the US Federal Size Regulations on commercial motor vehicles are seen for each dimension [14]. The minimum values for the dimensions of commercial motor vehicles can be taken as the maximum values for passenger vehicles. Vehicle speeds also depend greatly on the type of vehicle, as well as the type of road, weather conditions, and human nature. Figures 2 and 3 show the average speed recorded in 2012 in Great Britain for different types of vehicles on different types of roads [15]. Temperature readings can also help detect the presence of automobiles. The average operating temperature of a passenger vehicle is 90-105°C, while the surface temperatures are marginally lower [16]. As for emissions, automobiles produce CO_2, CH_4, CO, N_2O, and HFCs during operation [17]. Both petrol and diesel engines produce the same kinds of emissions, but at different amounts. Petrol engines produce more CO_2 and CO, while diesel engines produce more N_2O and other nitrogen oxides. Given these four parameters of size, speed, temperature, and emissions, a CubeSat surveillance system could detect automobiles from LEO.

Table 1. The maximum and minimum dimensions for new passenger vehicles in Europe.

Dimension	Value
Height	1.41 m – 2.12 m
Length	2.70 m – 5.40 m
Width	1.48 m – 2.07 m

Table 2. The maximum dimensions for commercial vehicles under US Federal Size Regulations.

Dimension	Value
Height	4.27 m
Length	13.72 m for buses
	14.63 m for semitrailers not including the cabin
	17.38 m for semitrailer-trailer not including the cabin
Width	2.6 m

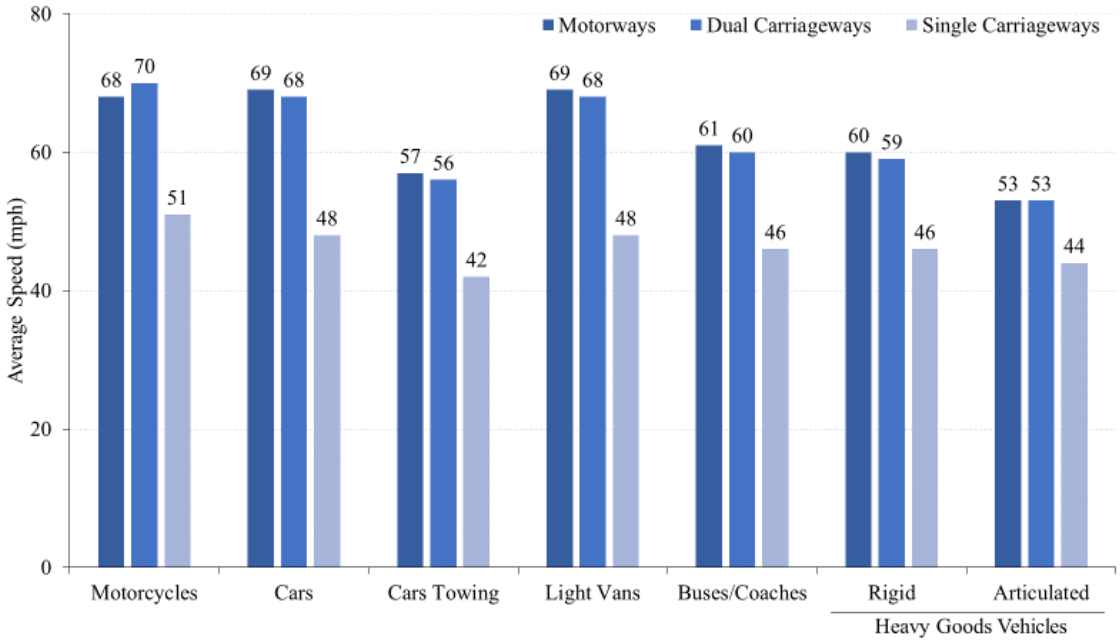


Figure 2. The average free flow speeds for different vehicles on different types of highways in 2012 in Great Britain.

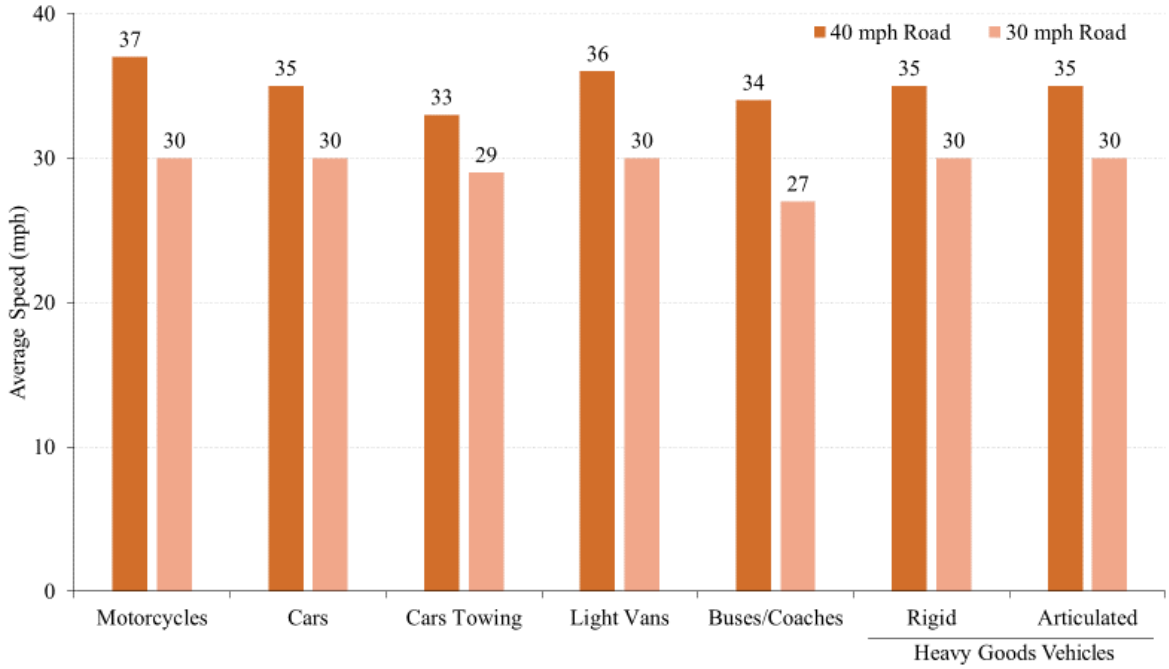


Figure 3. The average free flow speed for different vehicles on different types of city roads in 2012 in Great Britain.

As for airplanes, the CubeSat could monitor highjacked airplanes, or crash landings, for example. Like automobiles, there are many different types of aircraft with different dimensions. There are various commercial airplane manufacturers which produce different types of airplane models with varying parameters. For example, Boeing manufactures 17 different versions of its 737 model alone [18]. For military aircraft, designs vary greatly from country to country. Even within a single country, there exist different types of aircraft designed for different operations. Therefore, the parameters for the most common and for the biggest commercial aircraft were recorded, the Boeing 737-800 and Airbus A380-800. Also, the parameters for the most recognizable and for one of the biggest US military aircraft were recorded, the F-16 Fighting Falcon and C-5M Super Galaxy. The parameters included for the aircraft are size, operating altitude, cruising

speed, and engine exhaust temperature. Table 3 shows the size, operating altitude, cruising speed, and engine take-off exhaust temperatures for the most common type of passenger plane, the Boeing 737-800 [18,19,20]. Table 4 shows the size, operating altitude, cruising speed, and engine idle exhaust temperatures for the Airbus A380-800, the biggest passenger plane currently in operation [21,22,23]. Table 5 shows the size, operating altitude, top speed, and engine full thrust exhaust temperatures for the US Air Force F-16 Fighting Falcon [24,25]. Table 6 shows the size, operating altitude, and cruising speed for the C-5M Super Galaxy, which is one of the biggest aircraft used by the US Air Force [25,26].

Table 3. The size, operating altitude, and cruising speed for the Boeing 737-800.

Parameter	Value
Length	39.47 m
Height	12.55 m
Wingspan	35.79 m
Cabin Width	3.76 m
Max Flight Level	12496.8 m
Cruise Speed	949.513 km/h
CFM56-7 Engine Take-off Exhaust Temperatures	38-850°C

Table 4. The size, operating altitude, and cruising speed for the Airbus A380-800 and the engine exhaust temperatures for the two types of engines the model can use.

Parameter	Value
Length	72.73 m
Height	24.17 m
Cabin Height	8.41-8.56 m
Wingspan	79.8 m
Cabin Width	7.14 m
Max Flight Level	13136.88 m
Cruise Speed	1049.58 km/h
TRENT 900 Engine Take-off Exhaust Temperatures	40-956°C

Table 5. The size, operating altitude, and cruising speed for the F-16 Fighting Falcon.

Parameter	Value
Length	14.8 m
Height	4.8 m
Wingspan	9.8 m
Max Flight Level	15000 m
Cruise Speed	2414.016 km/h
F100-PW-229 Engine Full Thrust Exhaust Temperatures	37-1760°C

Table 6. The size, operating altitude, and cruising speed for the C-5M Super Galaxy.

Parameter	Value
Length	75.3 m
Height	19.8 m
Wingspan	67.9 m
Max Flight Level with a 605000 lb load	10363.2 m
Cruise Speed	870.7 km/h

1.1.2. Facilities and Infrastructural Emergencies of Interest

A key objective in the surveillance of the nuclear fuel cycle is the detection and characterization of facilities. The presence of a fuel fabrication facility in a remote and unauthorized location may be confirmed by the CubeSat surveillance system. Observing the facilities' building dimensions is a clear indicator of their presence. Since nuclear facilities of interest are likely to feature multiple buildings of different sizes and shapes, the surveillance system characterization methodology can focus more on typical features corresponding to certain facilities, like unique buildings or building layout. The Portsmouth Gaseous Diffusion Plant in Ohio among others were used as a reference the types of buildings and their dimensions at different nuclear facilities for the phenomena in this surveillance use example. Table 7 and 8 list the building dimensions for the biggest and smallest buildings of relevance for the Portsmouth Plant, the X-333 process building and X-600 steam plant, respectively [81]. Overall, building characteristics at all the plants in Figure 1 are of importance for the CubeSat surveillance system.

Table 7. The X-333 process building dimensions at the Portsmouth Gaseous Diffusion Plant.

Parameter	Value
Length	443.8 m
Width	295.7 m
Height	25.0 m

Table 8. The X-600 steam plant dimensions at the Portsmouth Gaseous Diffusion Plant.

Parameter	Value
Length	59.8 m
Width	30.3 m
Height	21.9 m

Structural fires and blackouts are two infrastructural emergencies that can pose a risk to human life if they happen at facilities along the nuclear fuel cycle. Fires or power outages at facilities could lead to the release of radioactive material to the environment. It is important to observe such events to provide adequate information on the situation to the appropriate authorities. Structural fires could be detected by the CubeSat system by looking at their temperature, gas emissions, and aerosol indices and optical depth. Fires which include structural materials burn at temperatures between 350-1200°C [27,28]. It is important for the CubeSat system to also make the distinction between structural fires and natural disasters. The temperatures of forest fires average at 800°C while the temperatures of lava flow and volcanic plumes range between 600-1200°C [29,30]. Although the sizes of structures vary, forest fires and volcanic eruptions generally take up a much larger surface area. As mentioned, the presence of structural fires could be detected by looking at its emissions when it burns. Gas emissions from structural fires include CO_2, CH_4, and NO_x, which come mainly from typical building materials [31]. These are the same gases as are emitted from forest fires; however, structural fires will also include the emissions of anything else within the structure that is burned, releasing additional types of gases that are not found in wildfires. Structural fire emissions can be distinguished easily from volcanic events on the other hand, since volcanic eruptions produce high amounts of SO_2

and no CH_4 or NO_x [30]. Apart from indicating incendiary events in structures, the same method of detecting gas emissions could point to other issues at facilities, for example a methane leak. Another method of detecting structural fires from space is looking at aerosol measurements from the smoke. The aerosol index could be used to compare the measurements of the level of aerosols present at the time of measurement versus the aerosol levels of a clean atmosphere [32]. An aerosol optical depth measurement could also be done, where a value of 0.5 and above indicates the presence of enough aerosols in the atmosphere to start blocking the sun [33]. As for blackouts, they could be detected by observing the absence of visible light, which is in the wavelength range of 400-700 nm. They could also be detected by an absence of thermal radiation, which has a typical bandwidth within 3-14 μm for most detectors [34].

1.1.3. Construction and Mining Events of Interest

The last phenomena that the CubeSat surveillance system will be observing for this use example are construction and mining activities. The presence or increase in both activities at facilities along the nuclear fuel cycle can indicate non-compliance of international safeguards. The CubeSat could alert to the unauthorized expansion of enrichment facilities by observing the presence of construction activities, for example. Also, observing the presence of mining could indicate malicious intent of a State attempting to acquire more nuclear material. Both phenomena have similar parameters for identification, such as the presence of equipment, temperature signatures, and gas emissions. Mining activities also include parameters like the presence of blasting agents and the mine surface area. For construction and mining equipment, Appendix A includes

tables with parameters for fifteen common vehicles that could be used for both activities [35,36,37,38,39,40,41,42,43,44,45,46,47]. The fifteen most common types of equipment are excavators, telehandlers, dragline excavators, bulldozers, tower cranes, compactors, trenchers, loaders, graders, pavers, wheel tractor scrapers, backhoes, feller bunchers, dump trucks, and pile boring equipment [35]. Table 9 shows an example of the information listed in Appendix A for the equipment. The temperature readings for both activities depend on the type of equipment and tools present. The CAT C4.4 ACERT engine in the CAT 320GC medium-sized excavator runs at a top tank temperature of 108°C [48]. Temperatures during any construction process involving concrete can reach up to 65°C [49]. The temperature of MIG welders can run anywhere between 6000-24000°C [50,51]. As for gas emissions, both activities produce CO_2, CH_4, N_2O, C_2H_2, and on rare occasions HFCs, PFC, and SF_6 from vehicle emissions, building materials, and onsite electricity consumption [52]. Mining activities also produce silica dust, which are quartz, cristobalite, or tridymite that makeup soil, granite, sand, and other minerals, and radioactive materials such as radon, ^{230}Th, ^{226}Ra, and all their decay products [53,54]. Unlike most construction activities, mining uses blasting agents to break open rock. The most common blasting agents used are nitroglycerine dynamite, and ANFO, which is a mixture of ammonium nitrate fertilizer and fuel oil [55]. Nitroglycerine dynamite explodes at 218°C and releases heat at 5000°C [56]. Another unique parameter for mining activities is the size of the mines. From above the Earth's crust, only open pit mines are visible. They usually have a surface area of 50-100 acres, which is the size of typical uranium deposits [55]. Open pit mines usually have a depth of 0-152.4 m [55]. Underground mine

networks are not usually visible from above the surface, but they reach depths of 91.44-304.8 m [55]. Since both construction and mining activities consume a lot of on-site electricity, the same methods for detecting the presence or absence of blackouts could be applied to detecting construction or mining.

Table 9. The parameters for the identification of an excavator. The values come from the CAT 320 GC medium-sized excavator.

Dimension	Value
Height to Top of Cab	2.96 m
Length	9.53 m
Width	3.17 m
Track Length	3.27 m

1.2. Application to Microreactor Surveillance

Although the main use example of the CubeSat surveillance system developed in this thesis includes phenomena of interest for nuclear surveillance, a smaller example is also explored in which the methodology is applied to the monitoring of microreactors. In a direct application, some of the methodology for sensors and data analytics developed for the CubeSat system can be applied to the monitoring of microreactor systems installed in remote locations. In an indirect application, the deep learning-based data analytics methodology can be adapted to different monitoring methods for microreactors.

While current power reactors in operation produce electricity at a magnitude of thousands of megawatts electric (MWe), microreactors are defined as units that produce less than 20 MWe [57]. According to the Department of Energy, microreactors are not representative of a certain fuel type or coolant, but instead have three main features:

factory fabrication, transportability, and self-regulation [57]. One major disadvantage of current light water reactor (LWR) designs is their lengthy construction costs and high capital costs. Microreactors aim to eliminate these issues in the nuclear industry by having all their components built, assembled, and integrated into a small unit at a factory that can then be transported to and installed at a different location. Due to the small size of the overall unit, the reactor could easily fit on the back of a truck and be taken to remote communities or military bases where there is not enough space or resources to allow for a conventional LWR or other power generation sources. Once on-site, microreactors would ideally self-regulate by requiring little maintenance, few to no operators, and feature a long core life.

Although microreactors present a very attractive solution for clean energy production in remote communities, their designs still present some issues. The biggest problem that must be overcome for the success of microreactors is ensuring the secure self-regulation or autonomy of the design. The most attractive feature for microreactors, remote operation, is also their biggest weakness given current regulations. Current reactor plants around the world have various security and safeguard measures implemented to help ensure global safety. Microreactors, on the other hand, cannot allow for such measures to be in place if they are operating in remote or potentially dangerous locations. For this reason, the strategies developed for global surveillance onboard the CubeSat system developed in this paper can be easily transitioned to create a solution for the microreactor monitoring issue. Like with the CubeSat platform, a system of sensors could be utilized in coordination with data analytics and artificial intelligence algorithms to

ensure the safety and security of microreactors in operation. Unlike the CubeSat surveillance system though, a global monitoring system for microreactors can also take advantage of on-the-ground sensors. Although there is a benefit to quickly observe and take measurements from LEO, accurate measurements of microreactor operation from ground sensors are needed in addition to increase the robustness of the monitoring system whose subject can pose a giant risk if it were to fall in the wrong hands. For example, sensors connected with the correct data analytics algorithms measuring the reactor's thermal profile and neutron flux could alert to a scenario where spent fuel was stolen from the microreactor. Although this example highlights a worst-case scenario for microreactor operations, it is not very probable since most microreactor designs feature integrated systems where all the components are installed inside the same containment unit. A more likely scenario, however, is the theft of an entire microreactor. Observing the presence or absence of a microreactor in its installation site can easily be achieved through a CubeSat surveillance system in LEO.

1.3. Thesis Objectives

As mentioned, the work in this paper establishes a foundation for a CubeSat-based surveillance platform for on-demand characterization of local phenomena. To reiterate, the aim of this thesis is to develop a methodology to characterize and identify diverse phenomena of interest among diverse objects in heterogeneous data sets for a nuclear surveillance use example. This overall goal of the thesis will be accomplished through the completion of the following objectives:

1. Development and compilation of representative surrogate data sets.
2. Conceptual development of a methodology for heterogeneous data analytics.
3. Illustrative applications of the methodology.

Although the work in this thesis does not culminate in the construction of an actual CubeSat system, the fulfillment of the objectives mentioned above will establish the mission goals and a base design architecture for the eventual physical development of the satellites and the data analytics methodology. The first objective of this thesis will prepare the data analytics structure of the CubeSat system for the recognition of the phenomena of interest. As with any artificial intelligence or machine learning problem, an adequate dataset is needed to successfully train the computer model so that it can be applied to new datasets. In other words, the CubeSat system must know what characteristics a phenomenon possesses prior to recognizing it. The surrogate dataset also aims at replicating the diversity in data a system with multiple different sensors collects. The second objective establishes the method which the CubeSat system will use for analyzing phenomena based on the data set constructed by the previous objective. Using the diverse types of data as input, the data analytics capabilities make predictions on the kind of phenomena the CubeSat system is observing. The final objective of this thesis elaborates on the applications of the work accomplished in the first two objectives. It illustrates how the data analytics model developed can be implemented in a realistic scenario. Before the completion of these objectives, extensive considerations were made about the orbital mechanics of the CubeSat system, satellite architectures and scenarios, and sensor selection for the system. It is important to maintain these considerations in mind as they can present limitations to the work accomplished under the three listed objectives. The

kind of data the CubeSat system can collect once deployed depends on the type of sensors onboard the satellites, and the sensors' capabilities for data collection depend on the satellites' architecture and behavior in orbit.

1.4. Thesis Overview and Methodology Development

To accomplish the objectives under this thesis, the paper is separated into eight total sections which explain in detail the work completed for the CubeSat-based surveillance system. Section 1 of this thesis serves as the paper's introduction. It presents the motivations behind developing the CubeSat surveillance system for the characterization of phenomena, and it describes in detail the different kinds of phenomena of interest for the specific use case. Defining these phenomena is the first step in developing the rest of the system. With the subjects of surveillance established, the system can be designed to meet surveillance needs for accurate characterization. This section also introduces the applications the CubeSat surveillance methodology can have towards implementing a monitoring system for microreactors, and it closes out by stating the thesis objectives and this thesis overview. Section 2 assesses the capabilities of a CubeSat-based surveillance platform. A definition for typical physical CubeSat architectures is provided, the satellite design process is explored, and the orbital mechanics for a CubeSat during deployment and orbit are analyzed. Simulations of a CubeSat orbit were accomplished using the NASA's General Mission Analysis Tool (GMAT) software. Section 2 provides evidence that a CubeSat serves as a feasible platform for a global surveillance system. Section 3 continues the analysis of the CubeSat platform by exploring different system architectures. Justifications are made for the use of a satellite constellation instead of a

single satellite, and the ideal communication options for the system were specified. The section also defines the specific CubeSat deployment scenarios by developing an algorithm for calculating the CubeSats' deployment times. Section 4 branches away from satellite architecture considerations by defining the types of sensors needed for the surveillance system. The sensor selection in this section is accomplished by first analyzing the surveillance requirements for observing the phenomena of interest defined in Section 1. Then, sensor requirements for observing the signatures and current technological capabilities are explored. Section 5 introduces the creation of the surrogate dataset for the characterization methodology by considering the sensor recommendations made in the previous chapter. The dataset aims at developing simplified versions of satellite images which can be used to create a general characterization algorithm. Also, the section introduces the concept of machine and deep learning and the importance of using adequate data. Section 5 accomplishes the first objective defined for this thesis. Section 6 then expands on the work in Section 5 by discussing the development of the characterization methodology. The methodology's architecture, its optimization processes, and training are all defined. The results of the characterization methodology on the surrogate dataset are listed in this section. Also, a final illustrative application combining all the topics covered in this thesis is described. Section 6 completes the final two objectives defined for the thesis. Section 7 explores the possible applications of the methodologies developed for the CubeSat surveillance system for the monitoring of microreactors. Finally, Section 8, concludes all the efforts accomplished in this thesis and makes considerations for future

work. Figure 4 illustrates the design methodology used to develop the final CubeSat surveillance system concept in this thesis.

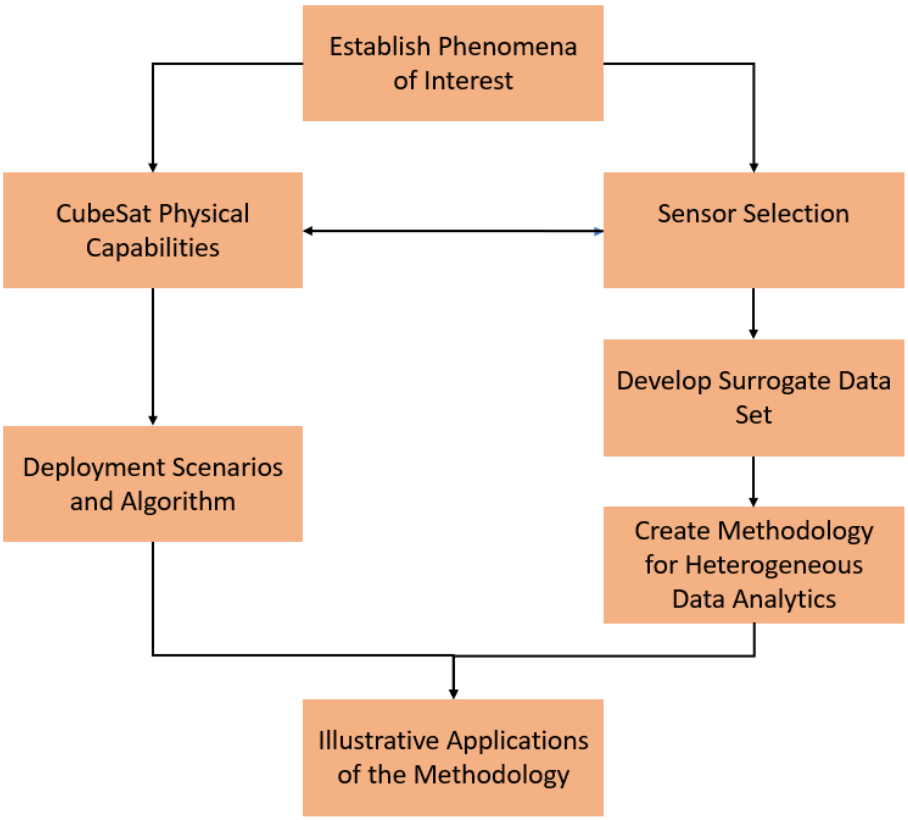


Figure 4. The design process used to develop the methodology for the CubeSat system.

2. CUBESAT-BASED SURVEILLANCE PLATFORM CAPABILITY ASSESSMENT

As seen in the previous section, various goals for the CubeSat surveillance system were defined. The CubeSat system must successfully observe multiple kinds of phenomena around the world to contribute to the surveillance of the nuclear fuel cycle. As previously stated, using CubeSats as a surveillance platform can allow for global coverage of phenomena on an on-demand basis in very little time. CubeSats are also an attractive option over other surveillance platforms due to their reduced cost and simplicity. Therefore, an important next step in the development of the method in this thesis is to assess the viability of a CubeSat system as a platform. This section will first provide a more specific definition of a cube satellite and explore its capacities for supporting a surveillance platform. Next, a summary of a general CubeSat design process is explored for facilitating future design since the work in this thesis only develops a method for the CubeSat surveillance system. To close out the section, a CubeSat's orbital capabilities in low Earth orbit (LEO) are examined through modelling.

2.1. CubeSat Description and Specifications

A big advantage for the use of CubeSats in this novel surveillance system instead of traditional satellites is their relatively low cost and ease of construction. CubeSats are simple enough in design that teams at universities should be able to develop their own satellites for their own scientific research, like the work being done for this thesis. There are many commercially-of-the-shelf (COTS) components available for CubeSats that facilitate the development of these systems by inexperienced teams [2]. As will be discussed later in this section, the CubeSats in this surveillance systems are not limited to

only the use of COTS components. There are multiple resources available that can aid any team in the development of their own CubeSats [2,3,59]. These resources outline step-by-step the processes necessary to constructing an operational satellite.

A CubeSat is defined as a miniature satellite of size 10 cm x 10 cm x 10 cm with a weight of around 1.33 kg as can be seen in Figure 5. This 10 cm cube is defined as a 1U CubeSat. Depending on the requirements of the mission and its necessary hardware, CubeSats can also take the shape of 2U, 3U, 6U, or even 12U, as can be seen in Figure 6. When taking previous CubeSat missions under consideration, it was found that 1U and 2U formfactors do not have enough space available within their structure to support the necessary sensors and hardware for Earth observation. To allow for the installation of high-resolution sensors on the CubeSats, the 3U and 6U form factors were considered. Due to the complexity of the various phenomena of interest for surveillance by this CubeSat system, more than one sensor is required to collect sufficiently diverse data for characterization. Since a CubeSat can only feature one to two onboard sensors because of their size, a single satellite is not sufficient for this surveillance system. Instead, a group of CubeSats, or a constellation, will form the system for the surveillance platform. Although the number of sensors was the biggest factor in selecting the number of satellites for the surveillance system, Section 3 compares the two system architecture scenarios in further detail. These multiple satellites can be launched from the ISS at the same time and will all adopt the same orbit for the rest of their lifetimes. Having numerous satellites allows the surveillance system to include multiple sensors and even have a certain level of redundancy with some sensors in case of failure.

Due to the diversity of sensors and their resolution requirements, the CubeSat constellation shall be made up of both 3U and 6U CubeSats. Although 3U CubeSats create less costs in development and launch, 6U CubeSats can house high-resolution sensors that exceed 3U dimensions. There also exists the possibility of developing unique sensors for surveillance in future work for this CubeSat system in order to meet resolution and size restrictions with either form factor.

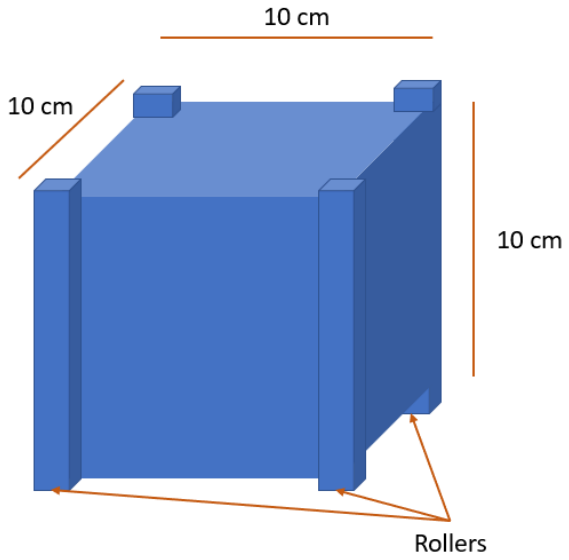


Figure 5. The dimensions of a 1U CubeSat.

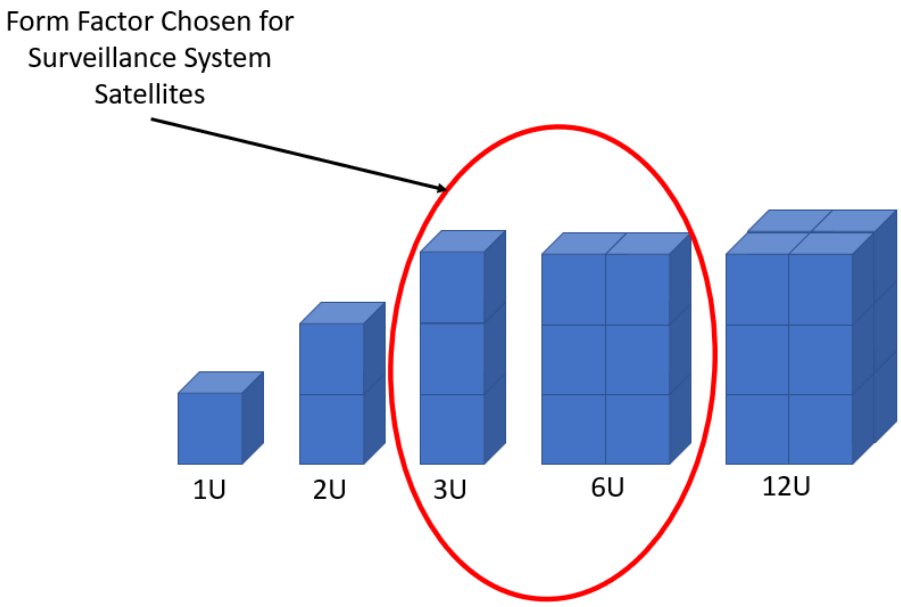


Figure 6. The different CubeSat formfactors.

Whenever CubeSats communicate to the surface, they utilize Radio Frequency (RF) transmission. Typical CubeSat transmissions operate in the VHF band, which is 30-300 MHz, in the UHF band, which is 300-3000 MHz, in the S-band, which is 2.0-4.0 GHz, and in the X-band, which is 8-12 GHz [2,78]. Most CubeSat missions take advantage of the UHF region for their RF communications since they have faster transmission rates than VHF, use less power than S-band or X-band signals. Signals in the UHF band are also easier and cheaper to license than S-band or X-band, which cuts time and costs in the CubeSat design process. Since most CubeSat mission deployed to date have mainly been experimental missions, they also did not need to take advantage of the higher security that comes from S-band or X-band transmissions. Due to their larger bandwidth, data sent through S-band or X-band can be easily encrypted. A typical UHF transceiver operates at up to 19.2 kbps, a typical S-band transceiver operates at 20 Mbps or higher and a typical

X-band transceiver operates at 150 Mbps [76,77,82]. Due to the nature of this CubeSat surveillance mission, the system will utilize signals in the S-band for uplinks and signals in the X-band for downlinks. Since the phenomena of interest for the CubeSat system involve emergency situations, the much higher communications speeds provided by S-band and X-band signals allow for faster transmission of time sensitive data. The higher security through encryption in the S and X-bands also provide an attractive advantage for the surveillance system. Utilizing two distinct frequencies for transmitting and receiving means that each CubeSat in the system will need an S-band receiver and an X-band transmitter instead of having just one transceiver. This S and X-band communications configuration also utilizes a lot more power than a UHF transceiver. In order for the transmitter and receiver power requirements to not interfere with sensor operations during observation, an adequate power duty cycle needs to be developed that accounts for the power consumption from all the components versus the power generation from the solar panels.

When considering CubeSats as a platform for a surveillance system, it important to recognize their lack of propulsion systems. CubeSats instead feature systems that can allow for rotation of the satellite while in orbit. Propulsion systems are usually out of the scope of most CubeSat developers due to their complexity, and since CubeSats are usually deployed from the International Space Station (ISS), they can pose a threat to the crew and ISS systems and hardware [3]. CubeSats, instead, can be rotated within orbit with either passive control systems, magnetorquers, or control mass gyros (CMG). Passive control systems involve using a magnet that takes advantage of the Earth's magnetic field

to always have the satellite facing towards the surface. An advantage of using passive systems is its simplicity and avoidance of power and software allocation. The issue with passive control systems, however, is a lack of precision in the satellite's orientation, which is a very necessary requirement for the level of Earth observation this surveillance system aims to reach [61]. Magnetorquers include magnets that can align with the magnetic field which allow for a CubeSat to rotate in any direction. They do not include any moving parts, which is advantageous for maintaining the integrity of the satellite. The most commonly used magnetorquers involve a tightly wound coil around a permalloy rod [62]. When a voltage is applied to the coil, a magnetic field is created, allowing the CubeSat to align itself with the Earth's magnetic field. CMGs feature a small mass that spins, creating enough angular momentum to rotate the CubeSat in any direction. These masses are usually in the form of momentum wheels or reaction wheels [62]. Although the wheels within a CMG provide a very high degree of orientation, or attitude, control precision, they can reach a limit of degradation after continued use. Some control systems, like the one proposed in [62], feature a combination of both magnetorquers and CMGs for high precision in attitude control and a longer lifetime. For this CubeSat surveillance system, an accurate and robust attitude control system is desired. Therefore, a system like the one proposed in [62] will be used.

Power supply systems also play a major role in analyzing a CubeSat's capability as a surveillance platform. Only sensors that fall under the power constraints for a CubeSat can be considered. Due to ISS safety restrictions, a typical CubeSat power supply produces 80 Wh [3]. There have existed special cases in the past which have included larger power

outputs, but they present a greater risk to the ISS and its crew. However, developing power supplies in-house with a larger power output may be necessary to accommodate certain sensors. The power system onboard a CubeSat includes several components: batteries, a power distribution unit, the charging circuit, and the solar arrays. A typical 3U/6U CubeSat solar pannel features 6-7 solar cells [63]. Each solar cell in a panel can produce ~1 W of power. Multiple panels can be connected via deployable solar arrays, allowing each array to contain 16-20 cells. The CubeSats for this system will feature two solar arrays, each containing 20 cells. Since the two arrays in conjunction produce 40 W of power, it would take two hours to charge a completely empty 80 Wh power supply. An adequate power duty cycle for each CubeSat must be developed that determines when each satellite component can operate to always have enough power available in the satellite. The high-power consumption components, like the transmitter, receiver, and sensor, should ideally not operate at the same time or while the solar panels charge the system. Taking the CubeSat's power output capabilities into consideration aid in sensor selection in future sections and an example duty cycle can be seen in Section 4.

As mentioned previously, many CubeSats in orbit today were launched from the ISS. The ISS was chosen as a launch platform for the CubeSats in this project for a variety of reasons. First, the launch provider collaborating on this project, NanoRacks, has their CubeSat launcher on the ISS. A graphic of their launcher can be seen in Figure 7 and Figure 8 [64]. The second reason for utilizing the ISS deals more with its orbital features. Given a CubeSat's restrictions on propulsion systems, its orbit cannot be changed to accommodate a certain target on the Earth's surface. Therefore, the CubeSat surveillance

system will have to be deployed at an orbit that allows it to eventually visit any point of interest on the surface. The ISS's orbit inclination of 51.6° means it can cover any point on the Earth between 51.6° N and 51.6° S in latitudinal coordinates, where most of the planet's population is located [9]. As can be seen from Figures 7 and 8, when a CubeSat is launched from the ISS, it is angled downwards and backwards from the front of the space station [64]. The spring system from the launcher gives the CubeSat enough velocity to avoid any collision with the ISS. Even though the CubeSat is launched away from the ISS, the launch velocity is not enough to overcome its initial momentum from when it was still docked on the ISS. Any CubeSats deployed from the ISS adopt the same orbit as the ISS, allowing for them to cover the same area. The ISS has a period of 92.8 min and can reach any point on the Earth within its inclination in around two to three days [9]. Having this type of orbit for a surveillance system is greatly advantageous as it would allow for quick access to any point of interest. When it is said that the CubeSat adopts the same orbit as the ISS, it is meant that the CubeSat adopts the same ephemeris data for the epoch time of when the satellite is launched. Once the CubeSat is in its orbit, it will slowly start deforming away from that of the ISS for a couple of reasons. First, the ISS regularly uses propulsion systems to maintain its orbit to fight against the effects of gravity and atmospheric drag, and as was mentioned earlier, a CubeSat is unable to use propulsion systems. Second, the difference in surface area and mass of the CubeSat, known as the ballistic coefficient, causes the CubeSat's orbit to decay at a quicker rate than that of the ISS [10]. The CubeSat orbital period becomes faster than that of the ISS because of its smaller semi-major axis of the elliptical orbit from launch. As the orbit decays, the semi-

major axis slowly gets smaller, which keeps increasing the orbital period [11,12]. Since a CubeSat is unable to regularly correct its orbit, its lifetime is usually around one to two years before it reenters the atmosphere [10]. Even though a CubeSat's orbit starts deviating from the ISS orbit, it will still cover the same amount of space as the ISS, just at a different rate. Another advantage for using the ISS as a launch platform is its ability to allow for "stash and deploy" methods. Unlike some other launch providers, a CubeSat on board the ISS can wait for its deployment until it is near an area of interest. This capability allows for the on-demand surveillance feature offered by this CubeSat platform.

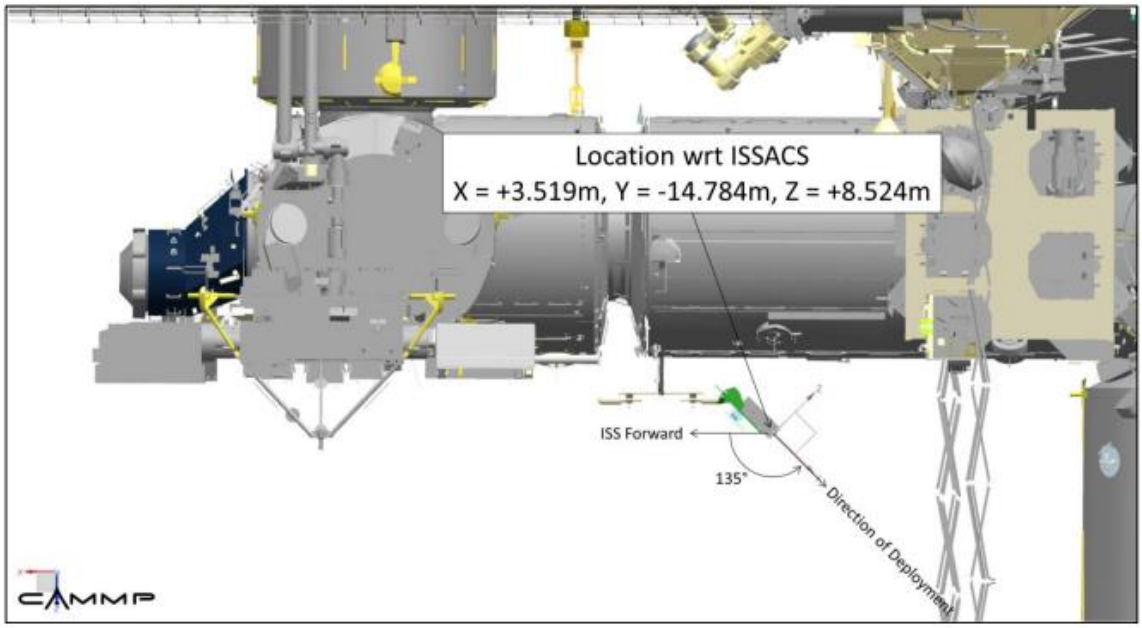


Figure 7. The NanoRacks CubeSat launcher attached to the ISS and its direction of deployment for the CubeSats. Reprinted with permission from [64].

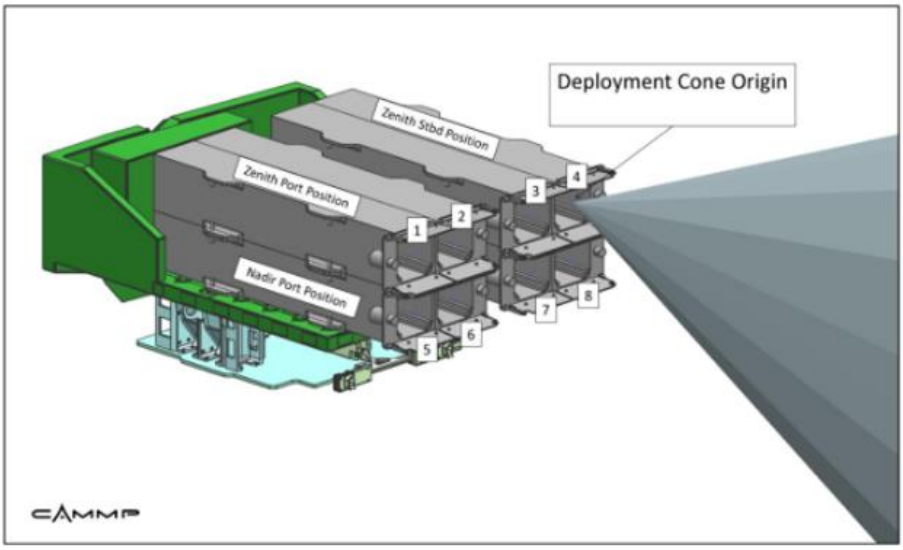


Figure 8. A Close-up of the NanoRacks ISS launcher and a cone showing deployment trajectory. Reprinted with permission from [64].

2.2. Satellite Design Process

Although the work in this thesis culminates in only a methodology for the CubeSat system, this subsection explores the process needed for the development of a mission ready satellite. Like with a typical CubeSat project, the design process for this CubeSat surveillance system is less complicated and expensive than for conventional satellites. The major factor contributing to the ease of designing a CubeSat is the availability of commercially off the shelf (COTS) components [2,3]. These COTS components allow design teams to concentrate on the payload and the final assembly of the CubeSat instead of having to design a transmitter from scratch, for example. The CubeSats for the surveillance system in this work have the capability of using COTS components for certain aspects of the satellite architecture, like onboard circuitry and solar arrays. However, due to the nature of this novel surveillance system, not all the components needed for the

CubeSat constellation can be purchased as COTS and would need to be developed in house. The determination of the use of COTS for this surveillance systems spurs from the needs and constraints for observing the phenomena of interest described in the previous section. For the CubeSats in the surveillance system to receive information on the various parameters characterizing certain phenomena, as stated in Section 1, a myriad of sensors must be placed onboard the satellite constellation which might not be commercially available. For example, an optical imager with a spatial resolution of a few meters could be used to identify the dimensions of construction vehicles. As is stated in Section 4, COTS imagers small enough to fit in the desired 3U form factor for the satellites in this surveillance system do not have the spatial resolution necessary [74]. However, the work in [68], for example, aims at developing an optical imager which fits inside a 3U CubeSat but can deploy from the satellite once in orbit in order to achieve spatial resolution of about 1.5 m from LEO, which is sufficient to identify construction vehicles. The kinds of sensors considered for use with this CubeSat system are discussed in Section 4, but the development of unique sensors for the platform is beyond the scope of this work. For the purposes of exploring the design process for the CubeSats in this surveillance system for this section, it is assumed that the sensors, or payloads, chosen will fit inside the CubeSat and allow for enough space for the other necessary components.

The rest of this subsection will explore the major steps and considerations needed for the design of this CubeSat surveillance system. There are multiple helpful resources available from different organizations, like NASA, that provide teams with guidelines and steps for the development of a CubeSat [2,3,59]. Using these guidelines, a more specific

definition for the satellites in this surveillance system is reached, but the work in this thesis does not aim at completing the design steps outlined in the guidelines. To illustrate how a CubeSat might look like for the surveillance system, multiple comparisons are made to the qbee50-LTU-OC (SE01) CubeSat which was part of the QB50 constellation deployed on 17 May 2017 [2,75]. The steps described in this subsection are merely the highlights of the CubeSat design process and does not intend to replace or replicate any of the very helpful design guides previously mentioned [2,3,59].

2.2.1. Getting Started

When beginning any CubeSat mission, the first things that need to be accomplished are developing a concept for the CubeSat and securing funding. It is easier to meet all necessary deliverables for producing a final CubeSat if there is always a goal in mind. Having a specific purpose for the CubeSat will determine what kind of hardware and licensing is necessary. As has been stated multiple times throughout this thesis, the objective for the CubeSats in this surveillance system is the characterization of phenomena of interest for nuclear fuel cycle surveillance. To recognize the phenomena specified in Section 1, various sensors are needed on board the CubeSat constellation. Therefore, the Cubesat architectures must allow for the functionality of such sensors and have the necessary equipment for data processing. When the design process begins for the CubeSat surveillance system, there needs to be extensive communication and collaboration between the CubeSat developer and the launch provider. A launch provider, like NanoRacks, is an organization that has the resources to place a CubeSat in orbit once it is completed. Within the developer team, specific tasks and responsibilities need to be

defined clearly for each team member. Appropriate delegation of tasks will ensure the success of developing the CubeSat(s). Once the design process for the CubeSat system begins, it is important to maintain a detailed timeline as many different aspects of the process should be developed simultaneously. For example, the CubeSat communications capabilities might affect the development of ground stations, and vice-versa. Testing and inspection of satellite components must also occur concurrently with the CubeSat development. As a comparison of what other CubeSat design processes look like, Figure 9 shows a timeline for the SE01 CubeSat mission's design process [2]. As can be seen, there are many different tasks needed to be completed for the successful development of a mission ready CubeSat, most of which happen simultaneously. The different tasks mentioned in the example timeline are described in the rest of this subsection. These same steps will be taken during the design process of the CubeSat surveillance system, but they will most likely not be the same duration. Factors like design, inspection, integration, programming, and testing will take longer to complete simply because the CubeSat surveillance system features multiple satellites instead of only one. Also, the task timeline assumes that the payloads, or sensors, for the CubeSats are fully developed or purchased and ready for integration.

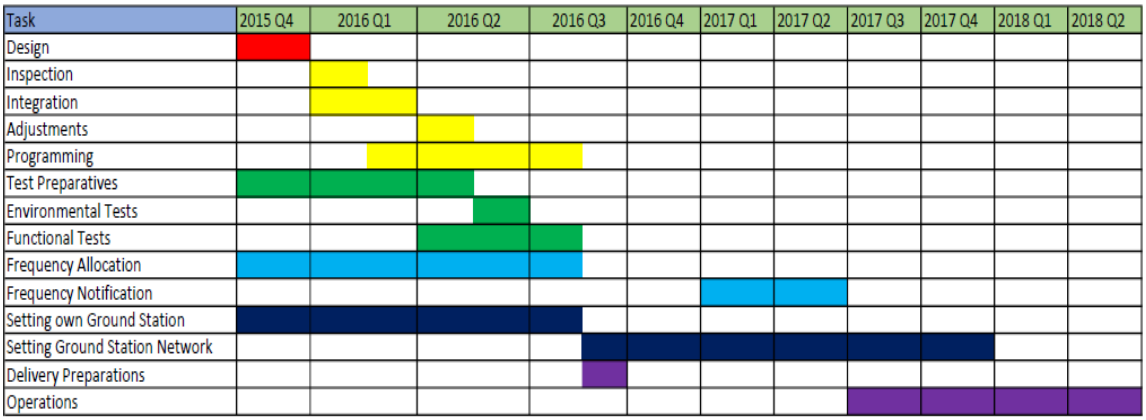


Figure 9. The timeline for the SE01 CubeSat Mission.

A very important aspect of the CubeSat design process which must begin at the start of the project is the licensing and frequency allocation of the satellite. This is a lengthy process that involves application submittals and communicating with regulatory commissions and can have an impact on the design of the satellite. If this portion of the development process is not completed before the scheduled launch time, the satellites are not allowed to launch and there could be even more serious repercussions. Therefore, it is recommended that this process is begun around 9 months to a year before the final delivery of the CubeSats to the launch provider [3]. In terms of licensing, all sensing systems in space require an issued license by the National Oceanic and Atmospheric Administration (NOAA) for operation. The process for obtaining a NOAA license begins by filling out an initial form which determines if a satellite mission needs a license. Once a need for a license is established, the licensing process will commence with NOAA. Once the developer team has the license for the CubeSats, frequency allocation can begin next. Once a CubeSat is in orbit, it communicates with ground stations by transmitting radio

frequency (RF) signals. Federal law requires that all radio frequency transmitters must be licensed before operation. The National Telecommunications and Information Administration handles all RF licenses for government operated satellites, while the Federal Communications Commission (FCC) takes care of any non-government operated satellites [59]. As for the FCC, there are three different paths developer teams can take: experimental, amateur, and commercial [3]. The type of path depends on the objectives of the CubeSat mission. The CubeSats for this surveillance system require an experimental license. Most CubeSats deployed by NanoRacks possess this license [3]. One last thing to consider for this CubeSat system is how the satellites' RF operations could interfere with the ISS. The Johnson Space Center Spectrum Office can analyze the possibility of interference between any satellite's RF signals and the ISS and should be contacted during the design process [3]. For more information on the licensing process, each agency should be contacted directly.

Before the fabrication and integration of the CubeSats begin, adequate facilities for the completion of each task must be established, especially for the development of non-COTS components. For the integration of components into the final satellites, a controlled environment, or cleanroom, is needed for meeting cleanliness protocols and avoiding unnecessary damage to the satellites. Adequate facilities are also needed for testing the CubeSats' ability to survive a space-like environment. The different types of tests are discussed in further detail later in this subsection. Finally, at least one ground station must be established for receiving communications from the CubeSats in orbit. As is discussed in Section 3, the CubeSats for the surveillance system in this thesis would

feature a network of different ground stations around the globe to increase the rate of data transmission during operation. The ground stations would feature antennas for communicating with the CubeSats, which vary depending on the type of RF signal the CubeSat transmitter operates on. A 2 m antenna would be needed for RF signals in the VHF band, which is 30-300 MHz, while only a 70 cm antenna is needed for UHF band signals, which are within 300-3000 MHz [2,78]. S-band signals between 2.0-4.0 GHz and X-band signals between 8.0-12.0 GHz are also commonly used and would require more of a dish or patch shape antenna [2,78]. The CubeSats in this surveillance system operate in the S-band region for uplinks and in the X-band region for downlinks to ensure greater communications speeds and security. Adequate power generators and computer interfaces are also very important to include in ground stations.

2.2.2. Satellite Components and Fabrication

As mentioned, the biggest advantage for the development of a CubeSat is its ability to use COTS components in its design. By using these types of components, the design and integration time of the CubeSat surveillance system can be accomplished quickly. As mentioned earlier, not all the components for the CubeSat surveillance system will be COTS, which can cause the design process to take longer than other CubeSat missions. In general, most CubeSats can be defined to include the following components: at least one payload, an onboard computer and data concentrator, a transmitter/receiver system, a power supply, an attitude control system, antennas, solar panels, and other external appendages [2]. For this CubeSat surveillance system, the components most likely not to be obtained commercially are the individual payloads, or sensors, and the power supplies.

Although there are multiple power supplies available as COTS, sensors developed in-house may need compatible power supplies. As for all the other components mentioned, it is completely feasible to purchase as COTS for this CubeSat surveillance system and should be done so. Table 10 lists all the components the CubeSats for this surveillance system should include and their viability to be purchased as COTS. It is important to consider at different steps in the design process that all components must be fabricated from appropriate materials that can withstand testing and meet the weight restrictions for CubeSat design specifications, which are 1.33 kg per 1U of size [2].

The first step in the design and integration of the CubeSat surveillance system is to determine the satellite's payload. The payload for the CubeSat, which is entirely dependent on the mission objectives, will determine all other aspects of the system. As previously stated, the payloads for the CubeSats in the surveillance system are the different types of sensors necessary for its observation tasks. Different types of sensors will require different sized CubeSat architectures, power supply sizes, transceiver types, and solar array sizes. The size of the satellite is something that should be considered concurrently with its type of payload. The CubeSat surveillance system will feature both 3U and 6U CubeSats. A payload should not be the exact same size as its CubeSat since other components also need to be included within the architecture. Depending on the sensor sizes, the CubeSats in this surveillance system will most likely feature only one payload per satellite. The CubeSat frames which will include all components of the satellites and forms their skeleton can be either bought or developed in-house if it meets the CubeSat

Design Specifications for the 3U and 6U form factors [58]. For visualization, Figure 10 shows what a 3U and 6U structure looks like [65].

Table 10. A list of CubeSat components for the surveillance system and their COTS viability.

Component	COTS viability
Payload or Sensor	Unlikely
Onboard Computer/Data Concentrator	Yes
Transmitter/Receiver	Yes
Power Supply	Likely
Attitude Control System	Yes
Antennas	Yes
Solar Panels	Yes
Other External Appendages	Yes
Structural Frame	Yes

Figure 10. Structural frames for a 3U and 6U CubeSat.

Once the payloads for each CubeSat are decided, adequate power systems should be chosen. As mentioned previously, typical CubeSat power supplies generate around 80

Wh of energy [3]. There have been special cases where this limit has been surpassed, but it comes at a higher risk of thermal runaway propagation. Even with power supplies below the 80 Wh threshold, tests to ensure its safety must still be conducted. Therefore, the power supplies for the CubeSats in the surveillance system will produce 80 Wh. Since the 80 Wh threshold is not exceeded, the power supplies can be purchased as COTS. If future work on the CubeSat system determines that a sensor with an extremely large power requirement is needed, the power supply for the satellite housing such sensor can be developed in-house. As mentioned earlier, the CubeSat power systems include batteries, a power distribution unit, a charging circuit, and solar arrays. Typical batteries used in CubeSat power systems are Panasonic Li-Ion 18650 batteries [66]. When selecting a power system, it is also important to consider power output and size requirements for the solar arrays. Although most solar arrays can deploy from the CubeSat once in orbit, they still cannot exceed CubeSat size requirements since they must be integrated within the CubeSat until its launch from the ISS. The CubeSats for this surveillance system will feature two solar arrays, each 20 cells producing a total of 40 W [63]. As previously mentioned, it will take the two solar arrays two hours to fully charge a completely empty 80 Wh power supply. As selections for power systems and solar arrays are finalized during the design process, it is important to consider the power budget of each component in the satellite to assure the CubeSats will indeed have enough power. Section 4 shows an example duty cycle for power consumption and generation.

For the next steps, the rest of the CubeSats' components should be acquired and integrated into the system. These include the attitude control system, the transmitter, the

receiver, its antennas, data concentrator, and onboard computer. As mentioned earlier, there are different options available for a CubeSat attitude control system. An attitude control system that allows for a wide range in motion with precision is desired for any CubeSats which observe the surface of the planet. An attitude control system like the one mentioned in [62] best serves the needs for the CubeSats in this surveillance system. In terms of power considerations, a typical CubeSat magnetorquer attitude control system requires up to 1.2 W during full actuation [80]. A transmitter/receiver configuration should be selected that can communicate in the required RF bands, which is determined in part by the licenses obtained from the FCC. As mentioned, the CubeSats will transmit X-band signals and receive S-band signals for increased speed and security in communications. A typical X-band transmitter consumes up to 12 W at maximum operations while a typical S-band receiver consumes up to 2 W [77,82]. The data concentrator and onboard computer should be adequately programmed to execute the desired mission objectives while the CubeSats are in operation. These objectives may include: the operation of the payload/sensor, allowing for communication between the satellite and ground stations, data processing and storage, etc. Since the main goal for the CubeSat surveillance system is to autonomously perform some onboard data processing for characterization before transmitting information to ground stations, the CubeSat onboard computers and data concentrators must be programmed to store the collected data and run the characterization algorithm. The methodology developed in this thesis aims at providing the basis for such programming. It is also worth noting that the CubeSats in this system will possess very high data storage capabilities in order to not limit data collection of phenomena before the

next transmission to ground. Typical onboard computers feature a power consumption of 400 mW on average [80]. The programming of the CubeSat components must be performed simultaneously with the satellite design and fabrication for seamless integration and time efficiency. As for the CubeSat antennas, they must be able to deploy from the satellite architecture after its launch from the ISS just like with the solar arrays as to not cause any issues during deployment. The antennas chosen for the system must be compatible with the transmitter and receiver RF signals. In terms of power consumption, typical X-band and S-band antennas requires 4 W for maximum RF output [79]. In order to utilize and integrate all on-board components of a CubeSat, multiple electronic boards are needed. Since CubeSats have very specific size restrictions, their circuit boards follow the PC/104 specifications [67]. To avoid issues like outgassing or whisker growing in the circuit boards, materials such as brass, zinc, and cadmium should be avoided [2]. Outgassing refers to a release of gases and particulates when material is within a vacuum environment, and whiskers are thin crystals that form from metals when in vacuum conditions and can cause short circuits [2,3]. As for wire insulation within the CubeSats, polytetrafluoroethylene (PTFE) or polyolefin (PO) should be used, not any polyvinyl chloride (PVC) materials [2].

A complete system layout for the SE01 CubeSat can be seen in Figure 11 as an example of how all the satellite components can be integrated within the structural frame [2]. The structure for the CubeSats in the surveillance system will resemble the SE01 layout but at a 3U/6U size instead of 2U. The payloads will be larger for the CubeSat system satellites, and they will most likely not feature a second payload. In terms of

materials for the CubeSat structure, aluminum alloys and austenitic stainless steel are typically used. Since the use of steel can create disturbances in the magnetic field, it should be used only for mechanical parts or fasteners [2]. During assembly of the satellite, it could become necessary to utilize bonding materials. Common types of bonding materials which are suitable for a space environment are epoxy, room-temperature vulcanizing silicone, and Kapton tape [2].

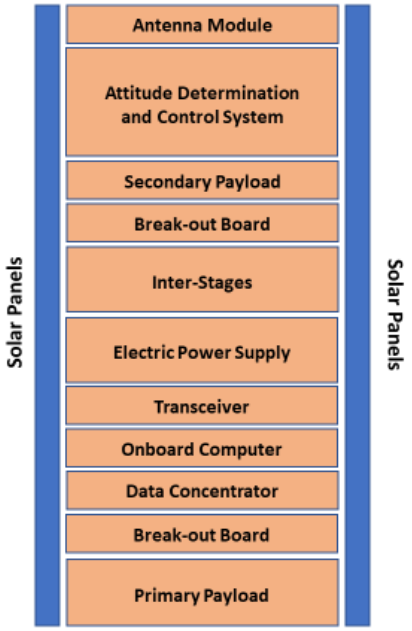


Figure 11. The final configuration for the 2U qbee50-LTU-OC (SE01) CubeSat mission.

Another aspect of the CubeSat design and fabrication process is the inclusion of inhibits on the satellite system. Inhibits must be included in the CubeSat architecture to prevent the satellite powering on prematurely. An unplanned powering up of the CubeSat can create inadvertent RF signals which can interfere with the RF systems of the launch

vehicle. In the case of CubeSats deployed from the ISS by NanoRacks, inhibits also help mitigate any risk to the crew [3]. For a mechanical implementation, inhibits could take the form of rollers or plungers which are in contact with the deployer's rails. Once a CubeSat is launched from the deployer, the inhibits disengage and the satellite is free to turn on. For an electrical application, inhibits can take the form of switches on circuit boards. In order to minimize a single-point failure, more than one inhibit should be integrated into the CubeSat.

2.2.3. Unique Component Considerations

Apart from the typical components found on a CubeSat, some missions in the past have featured unique components in their design that lead to special considerations. Some of these uncommon components with previous flight heritage which can be featured on the CubeSats for the surveillance system are higher power transmitters, deployable appendages, and lasers. It is important to consider the implication of implementing these unique features to the satellites since they may complicate the design and licensing process.

During the design process of the CubeSats, it may prove beneficial that a high-power transmitter should be integrated into some of the satellites to increase transmission speeds. Since typical transmitters already account for most of the power draw, a higher power transmitter may require a larger than average power supply. Radio Frequency signals present the biggest hazard during a CubeSat mission while on the ISS. The RF transmissions from a CubeSat could present a risk to the health of the crew and to critical ISS assets. It is extremely important to coordinate with the launch provider from the

beginning of the design process to make sure that the CubeSat's transmitter meets all requirements and that it will not come online until after deployment from the ISS. This is an example where inhibits in the satellite are crucial for mission success. As previously mentioned, the CubeSat surveillance system will use an X-band/S-band transmitter/receiver configuration. Although these components have high power consumptions, a power supply larger than 80 Wh will not be needed. Through licensing, inhibits, and coordination with NanoRacks, the X and S-band CubeSat components will be integrated into the satellites to avoid issues with the ISS and its crew.

Although any CubeSat must comply with the CubeSat Design Specification form factors, they can feature components that deploy beyond the CubeSats initial envelope. The most common types of deployable appendages featured on the CubeSat surveillance system are the solar arrays or sensors that extend from the bottom of the satellite. As previously mentioned, some types of imaging sensors might need to deploy and extend out of the CubeSat for operation [68]. For solar arrays, it is important to keep in mind that they do not have to necessarily deploy from the CubeSat. They can be installed on the surface of the satellite. Deployable solar arrays, however, can include a greater number of solar cells and power and can be moved to have a better angle for receiving the Sun's rays. As previously mentioned, all CubeSats in the surveillance system will feature deployable solar arrays to meet energy needs. Although deployable appendages are completely acceptable for a CubeSat, additional considerations must be made. The biggest concern with deployable components is that they may become caught by the deployer during deployment. These so called "hang fires" could create all sorts of collisions with the ISS

and present a danger to the crew [3]. To mitigate such danger, deployment testing is performed in coordination with the launch provider to ensure there is no possibility of a hang fire. If hang fires are prone to occur, the CubeSat design must be revisited.

It may become necessary for the CubeSats in the surveillance system as the design process begins to feature laser systems for optical communication capabilities or for sensors in Earth observation. Like with RF signals, the major concern for laser systems on the CubeSats is their interference with the ISS or its crew [3]. Therefore, they must be inhibited prior to launch from the ISS. Additional certifications and licenses must also be obtained to ensure the legal and safe operation of laser systems once in orbit.

2.2.4. Materials and Testing

Due to the hazardous and challenging environment at low earth orbit (LEO), any type of satellite under development must pass rigorous testing to ensure it operates correctly in space. As mentioned earlier, certain materials when placed in a vacuum are susceptible to certain side effects, like outgassing and whisker growth. The kinds of materials that should be used or avoided to mitigate outgassing and whisker risks have been previously mentioned in this subsection. In general, outgassing resistant materials can be chosen depending on two different properties: Total Mass Loss (TML) and Collected Volatile Condensable Material (CVCM) [3]. To meet NASA standards, the TML of a material should be less than or equal to 1% and the CVCM less than or equal to 0.1%. As a useful reference, NASA provides a database that includes the TML and CVCM of multiple materials [69]. It was also mentioned previously in this subsection to avoid the widespread use of steel as it could interfere with magnetic fields [2]. All materials selected

for the CubeSat design should also pass the ISS toxic material containment standards and re-entry survivability standards. A CubeSat waiting to be launched from the ISS should not present any toxicological hazards for the ISS crew, and all potentially toxic materials must feature sufficient levels of containment. Also, due to their size, the CubeSats should ideally burnup completely upon re-entry. If certain materials are not burned up because of high melting points, they must impart less than 15 Joules of energy at any point per FCC standards [3]. The ISS Program also requires adequate documentation for jettison assessment of such materials.

A major component for the CubeSat design process is the periodic testing of the satellites to ensure mission readiness. Without sufficient testing, the CubeSats will not be cleared for launch or in-orbit operations. There are three major kinds of tests which must occur during the design process: functionality, mechanical, and environmental testing. To test functionality, the CubeSats must be able to perform all required tasks in terms of software and hardware. It must be assured that the CubeSats can execute their functions and that there are no faults in the programming. Mechanical testing can be completed with the use of an electrodynamic shaker to test the CubeSats' ability to survive launch and deployment. Vibrations replicating a spacecraft launch can be programmed for the shaker. Testing of random vibrations on the CubeSats must also be conducted. The satellites' integrity must be unaffected, and all components must be functional for it to pass mechanical testing. Environmental testing involves thermal-vacuum bake-out and thermal-vacuum cycling tests to analyze the CubeSats' ability to withstand extreme temperature and vacuum environments [2]. During testing, it can be determined if

materials may experience issues like outgassing. It is also important to analyze if functionalities of the satellites are conserved after experiencing such environments. These tasks usually take between hours and days to complete and should therefore be scheduled accordingly.

2.3. Orbital Options and Modelling

The National Aeronautics and Space Agency (NASA) makes public the daily ephemeris data for the ISS which it updates regularly [70]. Given this data from NASA, the position and orbit of the ISS can be modeled with the correct software and algorithms. Table 11 shows an example of the ephemeris data posted by NASA for the ISS. As a first step for analyzing the orbital capabilities of a CubeSat, a proof of concept was created to explore if the ISS and CubeSat orbits do allow for access to any point of interest on the Earth's surface. Therefore, a model for the orbits of both satellites was desired. Figure 12 shows a free body diagram of the modelling process for the ISS and a CubeSat. As can be seen from the figure, the first step in the procedure for eventually calculating access times for a CubeSat, or how long the CubeSat is above a target on the surface, starts by first modelling the ISS orbit.

Table 11. An example of the ephemeris data provided by NASA for the ISS on February 28th, 2020 at 02:05:00.00.

Parameter	Variable	Value
Semi-Major Axis	a	6793034.89 m
Eccentricity	e	0.0009419
Inclination	i	51.67784°
Argument of Perigee	ω_p	81.96084°
Right Ascension of the Ascending Node	Ω	172.02147°
True Anomaly	ν	207.89816°
Mean Anomaly	M	207.94869°

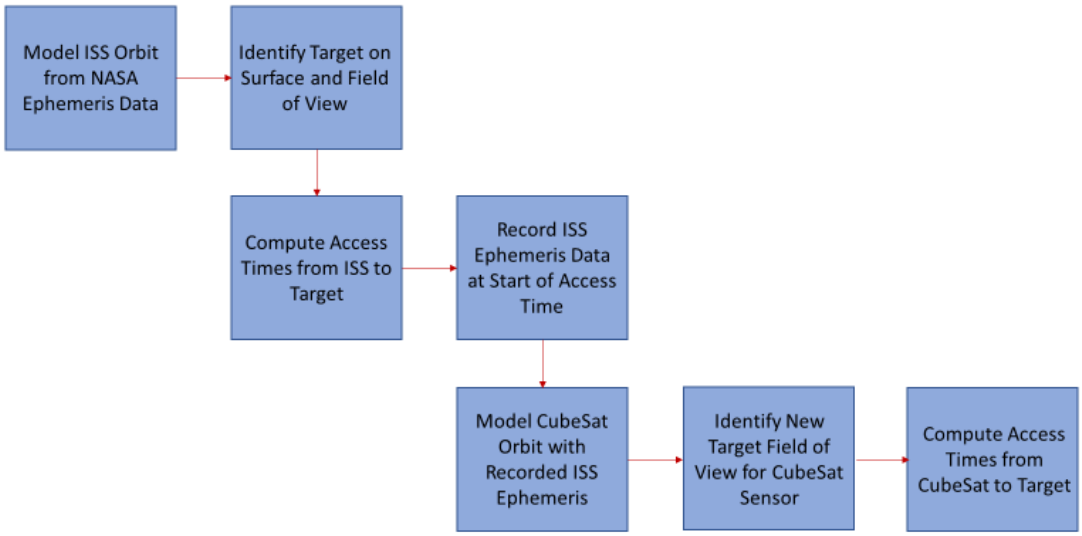


Figure 12. Steps for modelling satellites and computing the access time from the CubeSat to a target.

Initial attempts for creating a model for the space station's orbit involved using the Keplerian equations and Python code [11,12]. The python model of the Kepler equations used the ephemeris data of the ISS to calculate its current and future position in a latitudinal coordinate system. The developed model produced the latitude coordinate of

the ISS within 1° of error for up to a day in the future, but the model never accurately predicted the longitude coordinate. Since the Python model did not achieve the desired accuracy for modelling satellite orbits, it was decided that a higher fidelity model was needed to continue the work in this thesis. The use of software packages like STK by AGI and GMAT by NASA which feature complex propagators with very high fidelity was explored [71,72]. The package chosen to continue the work done in this paper was the General Mission Analysis Tool (GMAT) which is provided at no cost by NASA [72]. Figure 13 shows the interface for the GMAT software package. Given the live ephemeris data provided by NASA for the ISS, GMAT can model the orbit of the ISS. Figures 14 and 15 show the two different graphic displays that GMAT generates for modeling the ISS. The accuracy of the modelled orbit, however, only extends up to two to three days in the propagated model. Because of this, the model should be updated daily with the ephemeris data provided by NASA to maintain high fidelity in the calculations.

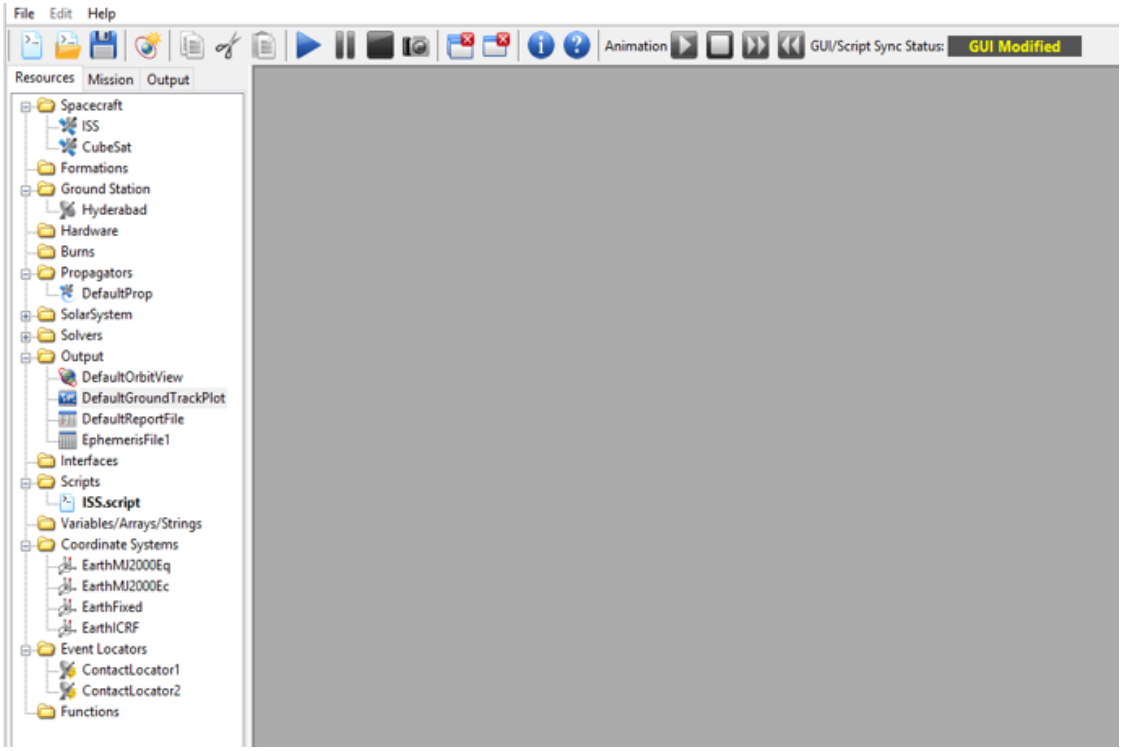


Figure 13. The interface for GMAT by NASA.

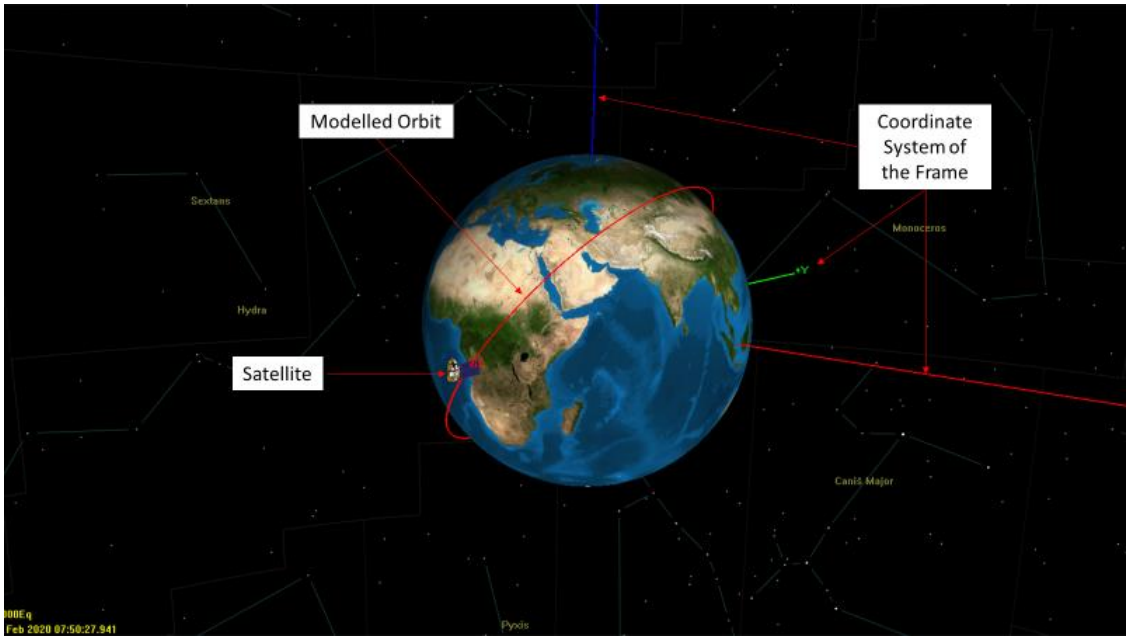


Figure 14. The planetary graphic display created by GMAT while modelling the orbit of the ISS.

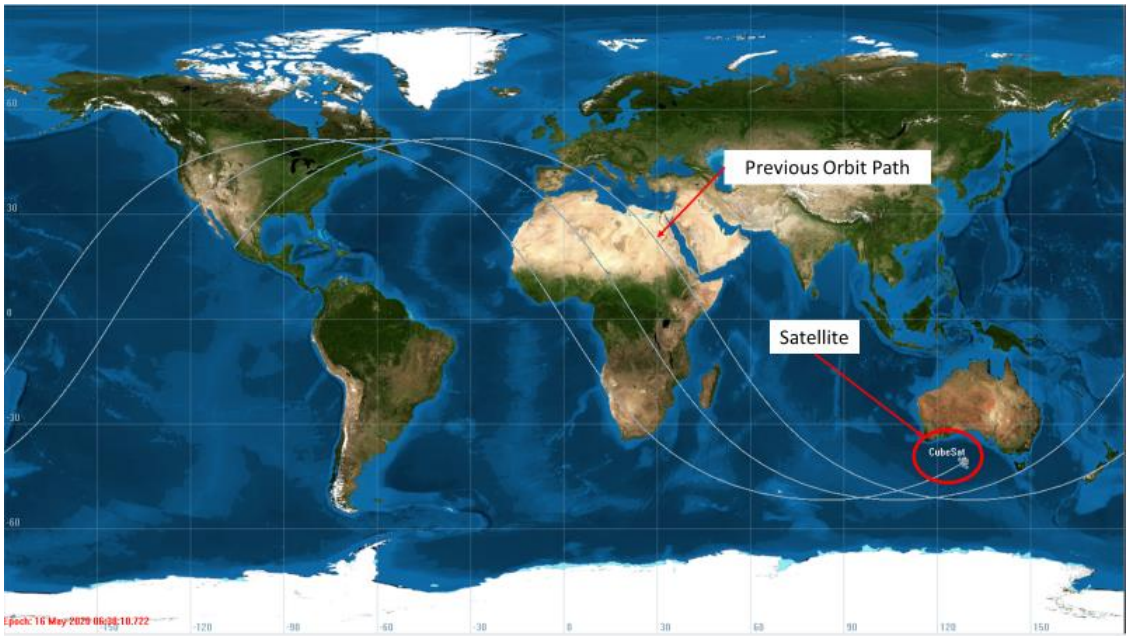


Figure 15. The Earth map graphic display created by GMAT while modelling the orbit of the ISS.

The steps for calculating the access times for a CubeSat to any target on the globe using GMAT as seen in Figure 12 are as follows. First, the ISS is modelled given the ephemeris data provided by NASA at the specific epoch. For example, if the data for the ISS provided by NASA was given at 12:00:00 pm on May 16th, 2020, the GMAT model will begin its propagation at the same time. As was mentioned earlier, the time window for the propagator must be updated for the model daily as well as the ISS's data in order to maintain a high accuracy. Once the ISS data has been input, a target location is placed on the surface given its latitudinal coordinates. The software propagator is then initiated, and the ISS access times are calculated for that location. Table 12 shows the output data GMAT produces for access times for the ISS. In this example, the city of College Station was the target, and the ISS experienced eight passes between May 16th, 2020 at

02:21:11.119 UTC and May 17th 2020 11:25:42.474 UTC. The access times for the ISS to a given location are calculated from when the ISS is coming just over the horizon and can see the target location. Whenever the access times for the CubeSat are calculated, they will be using a different angle of elevation. For imaging or sensing purposes, the CubeSat will need to be closer to the target for data acquisition, so the calculated access times reflect that. A graphic describing the angle of elevation from the surface to a satellite can be seen in Figures 16 and 17. The elevation angle is measured considering a vector at the 0° angle being on the horizon and the 90° vector being vertically up towards space from the surface. Therefore, a higher valued angle of elevation means less access to the location. For example, a 60° angle of elevation means a satellite can only see the location from the cone half angle between the 60° angle and the 90° angle. Once the access times are calculated for the ISS to a certain location, the ephemeris data for the ISS at the instant it begins the access time is exported and given to the CubeSat. The CubeSat is then modelled with the ISS ephemeris data with an epoch at the time of when the access time for the ISS begins. This is done in order to avoid orbit deformation as much as possible for at least the first pass of a location. This method of waiting until the ISS is above a target to deploy a CubeSat is what is called “stash and deploy”. When the ephemeris data is given to the CubeSat, its specific weight and ballistic coefficient are also added. Since the CubeSat is launched from the ISS in a downward direction, it will obviously begin with a lower semi-major axis. For the sake of the GMAT model, the semi-major axis is kept the same since a difference of a few hundred meters is negligible. Once the CubeSat is created with the necessary parameters, the model is propagated, and access times are calculated. Table 13

shows the access times for the CubeSat in the College Station example. The angle of elevation used for the data in Table 13 was 60°, which is a typical angle for satellite optical imagery [73]. As mentioned earlier, the access times for the CubeSat are completely dependent on the angle of elevation, which is dependent on the type of sensor used for observation. The type of sensor(s) used for this surveillance platform are explored later in this thesis. Once the type of sensor is determined, the CubeSat access times can be solidified. As can be seen from Table 13, the duration of the access times for the CubeSat to a location are not always the same. This is because the satellite will not always pass over a location at the same angle. Also, with a smaller angle of elevation, the CubeSat would have longer times and more instances of access times. The maximum time any location on Earth is visible from a satellite at a 500 km low Earth orbit (LEO) is about 11 min if the satellite is flying directly over the target. These 11 min will serve as an upper limit for deciding on the type of sensor that will be used. Sensors needing a longer time for data acquisition of a target will not be feasible for this surveillance platform.

Table 12. The start time, stop time, and total duration in seconds of the ISS access times to College Station between May 16th and May 17th, 2020.

Start Time (UTC)	Stop Time (UTC)	Duration (s)
16 May 2020 02:21:11.119	16 May 2020 02:28:50.439	459.320
16 May 2020 04:00:24.049	16 May 2020 04:03:58.692	214.643
16 May 2020 08:55:58.427	16 May 2020 08:59:33.090	241.663
16 May 2020 10:31:06.514	16 May 2020 10:38:46.245	459.730
17 May 2020 01:33:35.889	17 May 2020 01:40:56.842	440.953
17 May 2020 03:11:22.331	17 May 2020 03:16:58.536	336.205
17 May 2020 09:43:24.391	17 May 2020 09:50:51.132	446.742
17 May 2020 11:21:24.435	17 May 2020 11:25:42.474	258.039

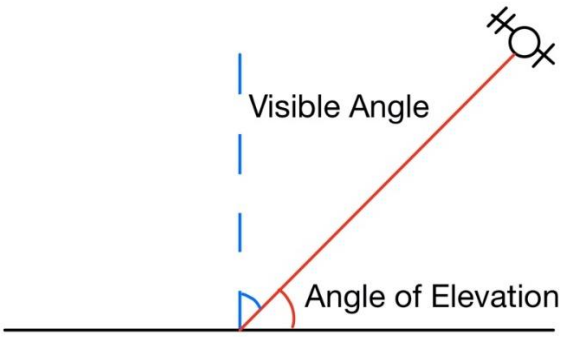


Figure 16. The definition of an angle of elevation with regards to a satellite.

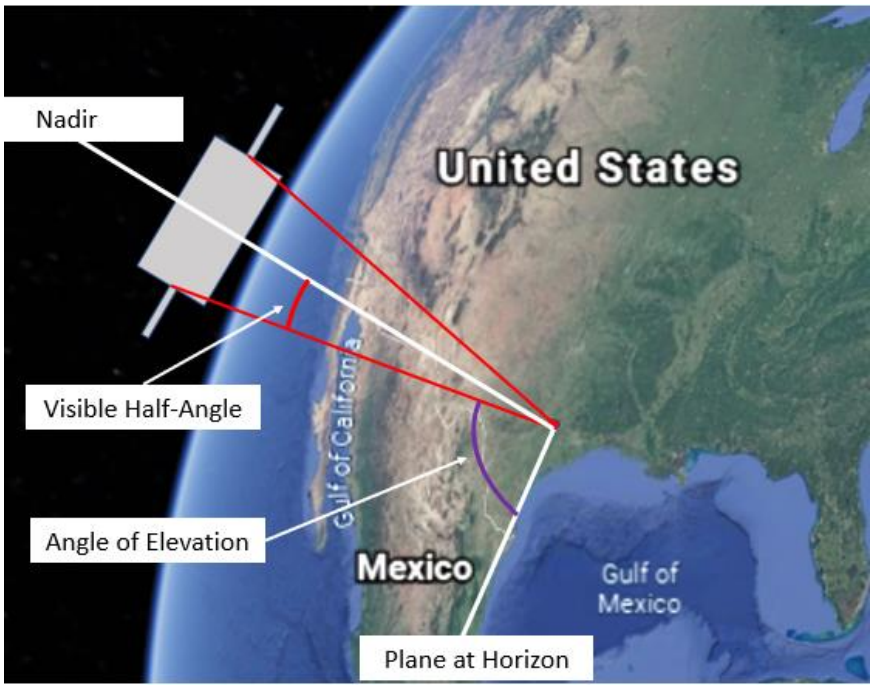


Figure 17. A 3D representation of the angle of elevation.

Table 13. The start time, stop time, and total duration in seconds of a CubeSat access times to College Station between May 16th and May 17th, 2020.

Start Time (UTC)	Stop Time (UTC)	Duration (s)
16 May 2020 02:24:28.108	16 May 2020 02:25:31.255	63.147
16 May 2020 10:33:28.336	16 May 2020 10:34:31.426	63.090

As can be seen by the access time calculation completed with GMAT, a CubeSat surveillance system launched from the ISS proves to be feasible. The CubeSat system can reach any point of interest on the Earth's surface and can do so more than once. The difference in the number of access times between the two satellites is due to the CubeSat's decaying orbit and its angle of elevation to the target. As was mentioned earlier, the types of sensors on board the CubeSat will affect the access times of the satellite to a location on Earth because of their needs for different angles of elevation. The less angle of elevation a sensor needs to collect data, the longer the access time duration will last. The amount of time available per pass for surveillance is a strong consideration for the type of observations that will be made by the CubeSat system.

2.4. Conclusion

This section assessed the viability of a CubeSat system as a platform to accomplish its surveillance goals. First, it provided a more specific definition of a cube satellite and explored its capacities for supporting a surveillance platform. The major components and capabilities of a CubeSat and the nature of an ISS deployment were defined. Next, a summary of a general CubeSat design process was explored for facilitating future. The design processes outlined in CubeSat designed guides were analyzed and adapted to the CubeSat surveillance system. Other CubeSat mission design procedures were also listed as reference points. To close out the section, a CubeSat's orbital capabilities in low Earth orbit (LEO) were examined through modelling. The General Mission Analysis Tool from NASA was used to observe the trajectory of a CubeSat's flight and analyze the kind of access it can have to targets on the surface

3. ARCHITECTURE AND DEPLOYMENT SCENARIOS

The previous section explored the capability of a CubeSat-based platform to support a global surveillance system. This section will expand on the information presented in Section 2 by comparing different possible architecture scenarios for the CubeSat surveillance system. It has been previously mentioned that a constellation of CubeSats proves most effective for accurate surveillance, but a closer comparison to a singular CubeSat surveyor is made to justify the use of a constellation. Different aspects of each system architecture were considered and compared to inform the final decision. Three different communications configurations for the CubeSat constellation were considered and analyzed in a decision matrix. Alternate system architectures for the constellation were also explored. Additionally, a mathematical algorithm was developed to define the deployment method of the CubeSat surveillance system from the ISS. With the completion of the deployment algorithm,

3.1. Comparison between 1 CubeSat and a Constellation

When asserting that a constellation of CubeSats is advantageous for global surveillance, a comparison must be made to a system featuring a singular satellite. Both options for the system present advantages and disadvantages. The most notable difference between these two architecture scenarios is the number of sensors the surveillance system can rely on. As previously mentioned in this thesis, 3U and 6U CubeSat structures only offer enough space for the integration of a single sensor. Therefore, since only one type of sensor is available for Earth observation in a singular CubeSat system, not all the previously listed parameters describing the phenomena of interest can be detected by one

satellite. For example, unless a new kind of sensor is developed, a singular CubeSat system cannot gather information about a structural fire's size, temperature, and chemical make-up with high enough individual accuracy for adequate characterization. Although a building fire can possibly be identified from LEO from only one of those parameters, there will be a much higher degree of uncertainty in characterizing the specific situation. If the system features an imager in the visible spectrum, it would only recognize the fire's size and nothing else. On the other hand, a constellation of CubeSats can feature as many sensors as there are satellites. Featuring a greater number of sensors allows the surveillance system to collect data of greater diversity when observing the target and increase its characterization accuracy. A constellation of CubeSats can permit the characterization of a structural fire's size, temperature, chemical make-up, and any other type of information about the event that can be gathered from other types of sensors onboard the system. A system with access to greater amounts of data allows the characterization method to more accurately analyze phenomena of interest. For example, since structural fires and wildfires have similar temperatures, having access to different types of parameters from the observed phenomena, like dimensional values or chemical make-up, allows the surveillance system to differentiate the two. Also, having numerous satellites allows for more than one sensor of the same type in the system. Redundancy with sensors in the system mitigates possibilities of sensor failure. Also, the presence of multiple satellites in the system increases its ability to survive an attack when compared to a single CubeSat system. If one of the satellites in the constellation were to be destroyed, the surveillance system as a whole would still function through the remaining satellites,

sparing any major communications interruptions experienced by the removal of one of the satellites in the constellation.

The next consideration is the system simplicity when comparing the singular CubeSat and constellation. Analyzing the simplicity of both architectures can be divided into two categories: constructional and operational simplicity. During the design and assembly phase of the CubeSat system, the singular CubeSat option has an obvious advantage in terms of number of satellites to be constructed. Less supplies and coordination are needed for the completion of a single CubeSat versus multiple. The diversity in sensors for a constellation greatly reduce simplicity as each individual satellite must accommodate a different sensor, especially in terms of power demand and software integration. On a licensing aspect, acquiring a license for a single satellite transceiver is less time consuming than for multiple transceivers. Also, having more satellites in the system increases the duration of the CubeSat design process by a factor of the number of satellites in the system. The construction of more satellites also translates to higher costs for the surveillance system. The cost of launch is also increased for a constellation since the price is based on the number of satellites sent to orbit. Although the price is not necessarily proportional to the number of CubeSats in the constellation, prices still increase with each additional satellite. As for operating in orbit, the singular CubeSat system still features greater simplicity than a constellation. Multiple different access times to a target need to be calculated for each satellite in a constellation. The difference in sensors onboard the surveillance system also creates different angles of elevation for each satellite. This increases computational time and resources when compared to only

calculating a single access time for a surveillance system with one CubeSat. On the other hand, since each satellite in the constellation has its own access time, the overall system has an access time to ground station communications which lasts longer by a factor of the number of satellites when compared to a single CubeSat system. Finally, the biggest source of complexity for a constellation when compared to a single satellite is the different communications options for the system. For a single CubeSat surveyor, it can only transmit information directly to a ground station during its access time. As will be discussed in the next subsection, a constellation of satellites has the option of exchanging information only through a ground site or through intersatellite communication before transmitting to ground.

After considering the benefits and disadvantages for each system architecture, the CubeSat surveillance system for this work is chosen to be made up of a constellation of multiple satellites, as has been mentioned previously in the thesis. In the decision process, phenomena characterization accuracy and system robustness were valued higher than everything else. The final surveillance system architecture was chosen using Table 14.

Table 14. The advantages and disadvantages between a 1 CubeSat or constellation system.

Attribute	1 CubeSat	Constellation of CubeSats
Higher Characterization Accuracy Through Sensor Diversity		✓
Lower Cost	✓	
Increased System Security		✓
Robustness Through Sensor Redundancy		✓
Simplicity	✓	
Longer Overall Access Times to Ground		✓

In Section 1, it was stated that the biggest advantages of using a CubeSat-based platform for remote monitoring over conventional satellite systems was the platform's increased simplicity and reduction of costs. As can be seen in Table 14, simplicity and costs are the two disadvantages of a CubeSat constellation when compared to a single CubeSat system. It is important to reiterate that even a CubeSat constellation proves advantageous in those two attributes when compared to conventional satellites due to the CubeSat sizes and use of COTS components. Although the assembly of multiple CubeSats takes much longer than for just one, time and resources do not have to be spent on the development of individual components custom made for the system, as is the case for conventional satellites. Conventional satellites may feature additional components not included in CubeSats that can increase their constructional and operational complexity, like propulsion systems. When comparing costs, a typical 3U CubeSat with X-band communications and a COTS multispectral imager would cost around $237,290.00 [85,74]. Assuming a constellation is made up of five CubeSats, the total cost, including the price of launch with NanoRacks, would be less than $2 million. The actual price of the CubeSat system in this thesis will vary as more or less CubeSats are needed in the constellation and as adequate sensors are developed in-house instead of acquiring them as COTS. On the other hand, NASA plans on spending approximately $165.7 million on the launch of their Geostationary Operational Environmental Satellite-T (GOES-T) alone [86]. The GOES-T satellite will provide data and imagery for multiple high-accuracy meteorological measurements [86].

3.2. CubeSat Constellation Scenarios

As was justified in the previous subsection, the optimal system architecture for the CubeSat surveillance system involves the use of a constellation of multiple satellites. The constellation configuration sacrifices the lower cost and complexity of the singular CubeSat option for increased characterization accuracy, system security, sensor redundancy, and longer overall access times. As part of the increased complexity of a constellation, there are multiple configuration options for the system. Different scenarios were considered for the final constellation design and are discussed in this subsection.

The most important consideration for constellations of satellites is the method in which they communicate. Three separate methods for accomplishing communications between the multiple CubeSats in the system were explored. The first method involves each individual satellite communicating directly to a ground station. In this scenario, each CubeSat transmits its individual data collected to a ground station during an access time. The on-board computers of each satellite run only a portion of the characterization algorithm which relates to the type of data its sensor collected. The satellites would then transmit their individual results to a ground station where the final portions of the characterization algorithm produce a final result. Ground station exclusive communication for the surveillance system presents the simplest option when compared to the other communications configurations, but it adds additional steps in the flow of data which can increase the time it takes for completing full characterizations of phenomena.

The second available method for system communications involves the CubeSats transmitting data between each other before sending it to the surface. This option allows

for the data from different sensors to compile within the system where the complete characterization algorithm can produce a result. System autonomy is gained with intersatellite communications as the ground station directly receives the end result of the characterization algorithm, and it requires less steps in the data flow. Since data is shared between the satellites in this communications option, only one of the CubeSats needs to be in range of a ground station to transmit the information to the surface, reducing the time needed to receive results. Since each CubeSat in the constellation may have a different ballistic coefficient once in orbit, their individual trajectories will deform slightly from each other, causing some satellites in the system to have access to a ground station when others do not. On the other hand, a constellation with ground station exclusive communications can only transmit all its data when each satellite has an access time to the station. The downfall of intersatellite communication, however, is its high levels of complexity and possible failure. For successful communications to occur between any two satellites, a direct line of sight between the two needs to be present. For conventional, large satellites, a direct line of sight is easily achievable through attitude control and propulsion systems. The CubeSats in the constellation can create direct line of sights through their attitude control systems, but their lack of propulsion does not guarantee they will persist throughout the lifetime of the CubeSats. In fact, the difference in ballistic coefficients of the CubeSats will eventually cause the distance between them to increase to a point where direct line of sights are lost for the rest of their lifetimes. Also, inter-CubeSat communication adds operational complexity to the system as compared to a

ground station exclusive system since each satellite must be adequately programmed to transmit, receive, and store information from the other CubeSats.

The third available method has the benefits of inter-CubeSat communications while aiming to resolve the line of sight challenge by utilizing another system of satellites as relays. For example, the Globalstar Network is a network of larger satellites further away from the surface of the Earth that offer enough coverage for all the CubeSats to always have constant access to them [60]. Using a relay satellite system like the Globalstar Network allows for the CubeSat surveillance system to keep its speed and autonomy for producing the final characterization of phenomena throughout the lifetime of the CubeSats similar to the inter-CubeSat communications option. Once the CubeSats in the system start drifting further from each other, communications between them are still possible since they will never lose coverage to the secondary satellite system. This communications option does present higher operational complexity when compared to a ground station exclusive option because of increased attitude control of the CubeSats to have a line of sight with the larger satellites and increased data transmission. Also, the use of a relay satellite system incurs an additional cost that comes with the service when compared to the other two options. Table 15 presents the decision matrix created when considering the three communications options. A green box represents a positive attribute; red, a negative attribute; and yellow, and intermediate attribute. It is important to note that the words in the boxes describe the attributes for the specific communications option while the colors describe the desirability. For example, “high” operational complexity is labeled as red since it is less favorable, while “high” speed to surface is labeled as green since it is more

favorable. When judging each aspect of the communications options, comparisons to a surveillance system with a singular CubeSat were also made. For example, even though the operational complexity of the ground station exclusive communications option is lower than the other two, it is rated as “moderate” instead of “low” since ground station exclusive communications for a singular CubeSat system are less complex than for the constellation.

Regardless of the communications method chosen from the ones previously mentioned, the data is eventually sent to the surface via a ground station. As previously mentioned, data transmission from a CubeSat to a ground station only occurs during an access time, making the communications rates largely dependent on the frequency of access times. For one ground station, the frequency of access times could vary between multiple times a day or have a wait time of up to two days before the next access time. If a certain phenomenon of interest involves an emergency which must be characterized in a short period of time, waiting up to two days for results is unfeasible. To overcome this obstacle, the transmission rate of data can be increased by allowing the CubeSat access to a network of ground stations around the globe. The Kongsberg Satellite Services (KSAT) provide a network of 23 station strategically placed all around the globe which any CubeSat mission to use [83]. Granting access to the KSAT network allows the CubeSat system to transmit data to the nearest station on its orbital path, greatly reducing the time between data collection by the sensors and final transmission to the surface. Even though only 17 of the 23 stations lie within a 51.6° inclined orbit, the CubeSat system

communications rate to the surface would still be 17 times faster when compared to a single ground station.

Table 15. The decision matrix for the CubeSat constellation communications options.

	Ground Station Only	Inter-CubeSat	Additional Satellite Relay
Operational Complexity	Moderate	High	High
Data Flow Steps	High	Moderate	Moderate
Speed to Surface	Moderate	High	High
Autonomy	Low	High	High
Lifetime	High	Low	High
Cost	Moderate	Moderate	High

Given the decision matrix above, the communications method most advantageous to the CubeSat surveillance system is the use of an additional satellite system further away from the surface, like the Globalstar Network, that acts as a relay for the data transmission between the CubeSats in the constellation. During the selection of the communications option for the system, different aspects of the constellation configurations were also considered. When analyzing the inter-CubeSat communications as an option, a solution to the loss of line of sight problem was considered in the form of tethers between the satellites in the constellation. By attaching cables between the CubeSats, they would never drift away from each other due to variable ballistic coefficients. A tethered constellation architecture configuration would increase the lifetime of the inter-CubeSat communications option and would make it the best communications option for the system when considering an updated decision matrix. However, a tether between satellites creates

more problems for the system than it solves. Section 2 mentioned that ensuring a successful deployment from the launch vehicle, the number of external components on a CubeSat should be minimal as to avoid any complications when deployed, like getting stuck in the deployer. The presence of an external cable attached from satellite to satellite increases this probability of a failed deployment. Even if a tethered constellation has a successful launch, other issues are created by the tether once in orbit. The metal cable could build up charge as it moves through the Earth's magnetosphere and potentially damage the CubeSats' hardware. Also, the presence of the tether in LEO creates additional drag with the atmosphere at that elevation, causing the orbital lifetime of the whole system to decrease. As is evident when comparing the advantages and disadvantages of a tethered constellation, this CubeSat surveillance system will not include one in its architecture.

Another alternative constellation configuration considered in this section involves dedicating one CubeSat in the system as the computational unit. This "computational" satellite within the constellation allows for all the data from the other CubeSats with sensors to compile in one location to run the final portions of the characterization algorithm before transmitting to ground. Without a dedicated CubeSat where the different types of data can be collected, data transmission between all the different satellites will resemble a complicated web, adding additional layers of complexity to how the characterization algorithm compiles data into a final answer. Additionally, since the "computational" CubeSat does not feature its own sensor, larger on-board computers may be installed to handle the large amount of data arriving from the other CubeSats and to increase computational resources for reducing the time the algorithm needs to run. One

downside of using the computational architecture option is that it leaves the CubeSat system more vulnerable to failure. If the "computational" CubeSat were to suffer a malfunction or some type of damage, the established data flow for the characterization algorithm would be interrupted. However, if that were to be the case, the CubeSat system architecture will resemble that of one which did not feature a "computational" satellite from the very beginning since each individual "sensing" CubeSat still features its own on-board computer. Re-programing of the system would be necessary, which could happen during the next possible access time of the system to a ground station. It is also worth noting that a computational satellite configuration is only feasible when using either the inter-CubeSat or the additional satellite relay communication option. Overall, this alternative constellation configuration increases characterization speeds and operational simplicity at the risk of reduced system robustness. After weighing the advantages and disadvantages, the CubeSat surveillance system will benefit from the use of a dedicated satellite within the constellation for demanding computational processes.

To summarize, Section 2 of this thesis explored the physical capabilities of a CubeSat and its individual components, while this section justified the specific operational architecture of the overall system. The CubeSat surveillance system will be made up of a constellation of satellites, each with a different sensor in order to increase the diversity of data collected and therefore the accuracy of the characterization algorithm. One of the satellites within the system will be used as a "computational" unit where all the necessary data will be compiled to produce a final characterization of the observed phenomenon of interest before transmitting to the surface. The CubeSats will communicate with each other

via an external satellite system further away from the Earth which will act as relays, like the Globalstar network. The system will also take advantage of the KSAT network of ground stations to increase the data transmission rate to the surface.

3.3. Deployment Algorithm

After the final CubeSat system architecture was defined, its deployment options were then explored. Although examples where given in Section 2 on how to model the ISS and CubeSat orbits with NASA's General Mission Analysis Tool (GMAT) for calculating access times, a mathematical definition for that process must be defined to illustrate the CubeSat system's deployment options. The algorithm developed in this section does not aim at replacing the complicated orbital mechanics calculations which GMAT handles. Instead, this algorithm mathematically describes when to launch the CubeSat surveillance system from the ISS and the duration of its access times. Figure 18, as seen in previous sections, lays out the different steps the algorithm aims to describe, and Figures 19a and 19b outlines the algorithm structure. The rest of the section describes in detail the steps in Figures 19a and 19b.

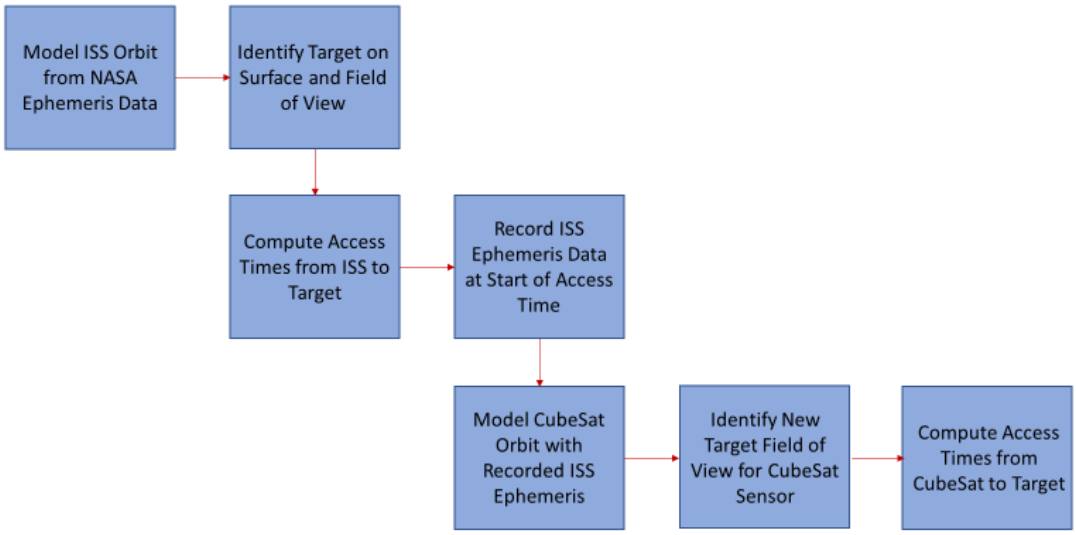


Figure 18. Algorithm flow for calculating CubeSat access times to a target.

As seen in Figure 18, the first step in the process is to model the ISS's orbit. Since the orbital modelling is done through GMAT, the ISS's orbit can be mathematically described by Eq. 3-1 to illustrate the variables of interest to the user. In Eq. 3-1, the ISS's position at a given time in cartesian coordinates is described by its orbit function O_{ISS}, which is a function of time and the ISS ephemeris data at epoch. For reference, an epoch refers to an initial starting point in time. As previously discussed, the ephemeris data of the ISS is made public by NASA at daily epochs. For example, the ISS ephemeris data for June 3rd, 2020 was recorded and published at 23:25:30.983 GMT (that day's epoch).

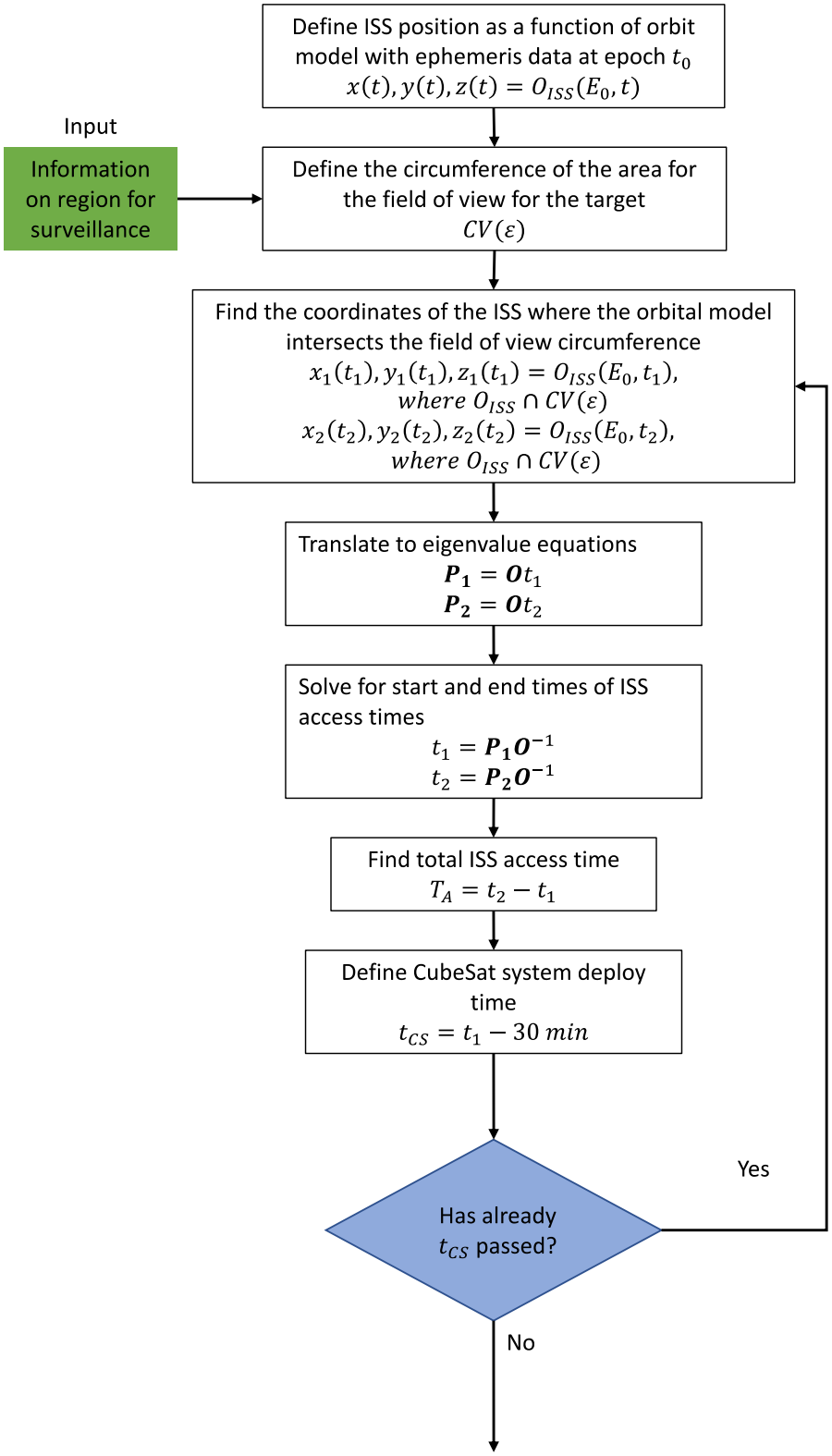


Figure 19. Algorithm structure to evaluate deployment characteristics.

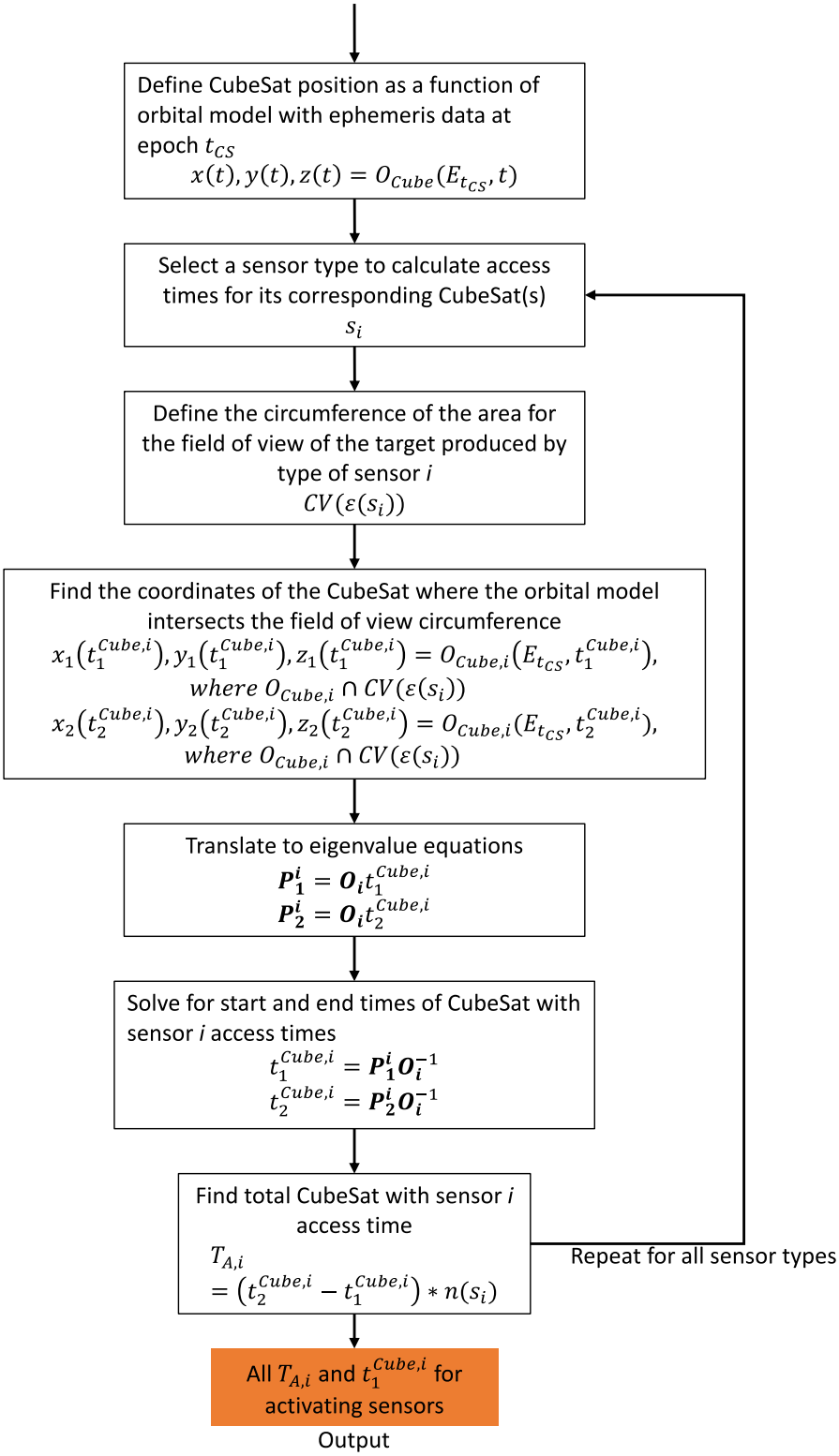


Figure 19. Continued.

$$x(t), y(t), z(t) = O_{ISS}(E_0, t) \tag{3-1}$$

Where,

$x(t), y(t), z(t)$ = The ISS cartesian coordinates at time *t*

$O(E_0, t)$ = The ISS orbital model function at time *t* dependent on ephemeris data E_0 at epoch time t_0

With Eq. 3-1 describing the ISS's position at any given time, access times for the ISS can be calculated to any target on the surface. First, a target must be identified by defining its coordinates. Usually, it is easier to find a target's coordinates in latitudinal coordinates first and then translating to cartesian. For example, the latitudinal coordinates for the city of College Station are 30.6280°N and 96.3344°W at an altitude of 103 m. Although the target may be a single point, the ISS's view of the certain target occurs over an area. The ISS's position during its access times to a target can be described by Eq. 3-2 and 3-3. The ISS has access to a target between position 1 and 2 within the field of view as can be seen in Figure 20. The area around a target defining the field of view is dependent on the satellite's angle of elevation to the target, as described in detail in Section 2. A smaller angle of elevation creates a larger view angle which increases the visible area. A typical angle of elevation for the ISS to start having access to a target is around 6-8° [84]. The radius for the field of view is therefore the distance from the target to a height of 6-8°. The coordinates for Position 1 and 2 are where the orbit intersects with the circumference of the field of view area and are described in terms of the ISS orbit in Eq. 3-2 and 3-3.

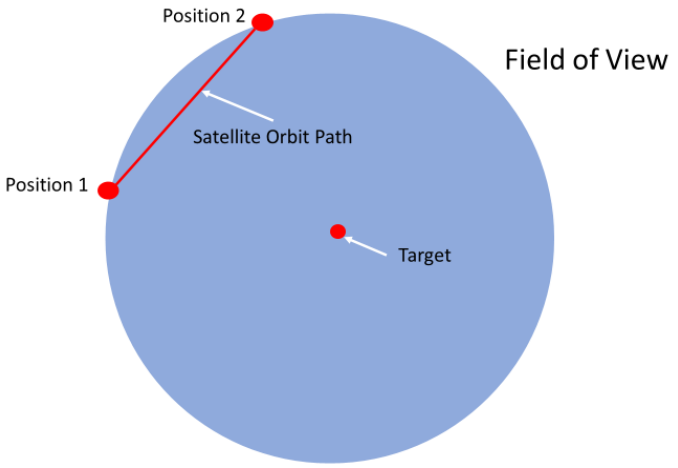


Figure 20. An example of a satellite pass over a field of view of a target.

$$x_1(t_1), y_1(t_1), z_1(t_1) = O_{ISS}(E_0, t_1), \quad where\ O_{ISS} \cap CV(\varepsilon) \tag{3-2}$$
$$x_2(t_2), y_2(t_2), z_2(t_2) = O_{ISS}(E_0, t_2), \quad where\ O_{ISS} \cap CV(\varepsilon) \tag{3-3}$$

Where,

$x_1(t_1), y_1(t_1), z_1(t_1)$ = The position (Position 1) of the ISS at the beginning of its access time to a target at time t_1

$x_2(t_2), y_2(t_2), z_2(t_2)$ = The position (Position 2) of the ISS at the end of its access time to a target at time t_2

$CV(\varepsilon)$ = Circumference of the field of view as a function of angle of elevation ε.

With the intersection locations defined, the times of intersection and total access time for the ISS is found. Eq. 3-2 and 3-3 can be translated into eigenvalue equations transforming Eq. 3-1 into Eq. 3-4 with *t* as the eigenvalue. The transformed Eq. 3-2 and 3-3 can be seen in Eq. 3-5 and 3-6.

$$\boldsymbol{P} = \boldsymbol{O}t \tag{3-4}$$

Where,

t = The eigenvalue representing time

$\boldsymbol{P}$ = The eigenfunction for position of the satellite

$\boldsymbol{O}$ = The eigenfunction for the orbit model

$$\boldsymbol{P_1} = \boldsymbol{O}t_1 \tag{3-5}$$

$$\boldsymbol{P_2} = \boldsymbol{O}t_2 \tag{3-6}$$

Where,

t_1 = Time when access time starts
t_2 = Time when access time ends
$\boldsymbol{P_1}$ = The position of the ISS at the beginning of its access time to a target at time t_1
$\boldsymbol{P_2}$ = The position of the ISS at the end of its access time to a target at time t_2
$\boldsymbol{O}$ = The ISS orbital model function, still dependent on ephemeris data E_0

The start and end times, t_1 and t_2, for the ISS access time to the target can be found by inverting and multiplying $\boldsymbol{O}$, as seen in Eq. 3-7 and 3-8.

$$t_1 = \boldsymbol{P_1}\boldsymbol{O}^{-1} \tag{3-7}$$
$$t_2 = \boldsymbol{P_2}\boldsymbol{O}^{-1} \tag{3-8}$$

The total access time duration can then be found with Eq. 3-9. It is important to keep in mind that the mathematical algorithm is only a representation of the calculations done by GMAT. As mentioned earlier, GMAT will produce all access time instances and durations for a satellite given its ephemeris data, angle of elevation, and a target location.

$$T_A = t_2 - t_1 \tag{3-9}$$

To reiterate, t_1 is the time at which the ISS crosses the target's field of view. Whenever CubeSats are deployed from the ISS, they must wait a total of 30 min before they power on in order to avoid their RF signals interfering with ISS systems and signals. Therefore, in order to have an operational CubeSat system once its orbit reaches the target, it time of deployment must be defined by Eq. 3-10.

$$t_{CS} = t_1 - 30\ min \tag{3-10}$$

The CubeSat system will adopt the same ephemeris data as the ISS at time t_{CS} in the ISS orbit, represented as $E_{t_{CS}}$. If the calculated time for t_{CS} has already passed at the time of calculation, Eq. 3-2 – 3-10 must be redone for the next pass of the target in the ISS orbit. If t_{CS} is still in the future, the algorithm can be continued. As with starting to model the ISS, the CubeSat system's orbit can be mathematically described by Eq. 3-11. In Eq. 3-11, the CubeSat system's position at a given time in cartesian coordinates is described by its orbit function O_{Cube}, which is a function of time and the CubeSat ephemeris data at epoch t_{CS}.

$$x(t), y(t), z(t) = O_{Cube}(E_{t_{CS}}, t) \tag{3-11}$$

Like the process with the ISS, the next step is determining the coordinates of intersection between the CubeSat orbit function, O_{Cube}, and the circumference of the field of view for the CubeSat, $CV(\varepsilon)$. However, the elevation angle, ε, for the CubeSat will be different to that of the ISS. The CubeSat's angle of elevation depends on the type of sensor(s) on board. Some sensors require a more direct line of sight to the target in order to collect data, making the area for the field of view much smaller. Therefore, the circumference of the field of view for the CubeSat can be rewritten as $CV(\varepsilon(s_i))$, making the elevation angle dependent on the type, *i*, of sensor, *s*, onboard a specific CubeSat. For CubeSats in the constellation with other sensors, different circumference functions must be considered since their field of view is different. For example, $CV(\varepsilon(s_j))$ represents the circumference of the field of view for a CubeSat in the constellation with sensor type *j*. The coordinates for where the CubeSat with sensor type *i* intersects the circumference of the field of view for the target are represented by Eq. 3-12 and 3-13. Also, since access

times are specific to each CubeSat in the constellation, the orbital model function is changed to $O_{Cube,i} = (E_{t1}, t)$ to represent the orbit for any CubeSat in the constellation with sensor type i.

$$x_1\left(t_1^{Cube,i}\right), y_1\left(t_1^{Cube,i}\right), z_1\left(t_1^{Cube,i}\right) = O_{Cube,i}\left(E_{t_{CS}}, t_1^{Cube,i}\right),\ where\ O_{Cube,i} \cap CV(\varepsilon(s_i)) \quad (3\text{-}12)$$

$$x_2\left(t_2^{Cube,i}\right), y_2\left(t_2^{Cube,i}\right), z_2\left(t_2^{Cube,i}\right) = O_{Cube,i}\left(E_{t_{CS}}, t_2^{Cube,i}\right), where\ O_{Cube,i} \cap CV(\varepsilon(s_i)) \quad (3\text{-}13)$$

Where,

$x_1\left(t_1^{Cube,i}\right), y_1\left(t_1^{Cube,i}\right), z_1\left(t_1^{Cube,i}\right)$ = The position of the CubeSat with sensor *i* at the beginning of its access time to a target at time t_1

$x_2\left(t_2^{Cube,i}\right), y_2\left(t_2^{Cube,i}\right), z_2\left(t_2^{Cube,i}\right)$ = The position of the CubeSat with sensor *i* at the end of its access time to a target at time t_2

Like with the ISS, Eq. 3-12 and 3-13 can be transformed into an eigenvalue representation using Eq. 3-4. The transformations are seen in Eq. 3-14 and 3-15.

$$\boldsymbol{P_1^i} = \boldsymbol{O_i} t_1^{Cube,i} \quad (3\text{-}14)$$

$$\boldsymbol{P_2^i} = \boldsymbol{O_i} t_2^{Cube,i} \quad (3\text{-}15)$$

Where,

$t_1^{Cube,i}$ = Time when access time starts for CubeSat with sensor *i*

$t_2^{Cube,i}$ = Time when access time ends for CubeSat with sensor *i*

$\boldsymbol{P_1^i}$ = The position of the CubeSat with sensor *i* at the beginning of its access time to a target at time t_1

$\boldsymbol{P_2^i}$ = The position of the CubeSat with sensor *i* at the end of its access time to a target at time t_2

$\boldsymbol{O_i}$ = The CubeSat with sensor *i* orbital model function, still dependent on ephemeris data E_{t1}

The start and end times for the CubeSat with sensor *i* access time to the target can be found by inverting and multiplying $\boldsymbol{O_i}$, as seen in Eq. 3-16 and 3-17.

$$t_1^{Cube,i} = \boldsymbol{P_1^i O_i^{-1}} \tag{3-16}$$

$$t_2^{Cube,i} = \boldsymbol{P_2^i O_i^{-1}} \tag{3-17}$$

The total access time for CubeSat with sensor *i* can then be found by Eq. 3-18. Since a constellation may have more than one CubeSat with the same sensor, the access time is multiplied by the number of CubeSats with sensor $i, n(s_i)$. The steps represented by Eq. 3-12 – 3-18 can be repeated for CubeSats in the constellation with different types of sensors.

$$T_{A,i} = \left(t_2^{Cube,i} - t_1^{Cube,i}\right) * n(s_i) \tag{3-18}$$

Using the mathematical algorithm developed and the corresponding GMAT calculations for the CubeSat system to identify its access time duration and start time to a target, a decision can be made about deploying the CubeSat system for the observation of a target. Deployment of the surveillance system is initiated at the calculated access time start time, and the sensors will be activated for the calculated duration of the start time. In the field, it is recommended to activate the sensors for a longer period than the calculated access time duration to mitigate orbit calculation errors. As an example, Figure 21 describes the data flow for the CubeSat surveillance system observing and identifying suspicious activity at an enrichment plant. Figure 21 illustrates how different algorithms and processes are integrated for a final CubeSat system operation.

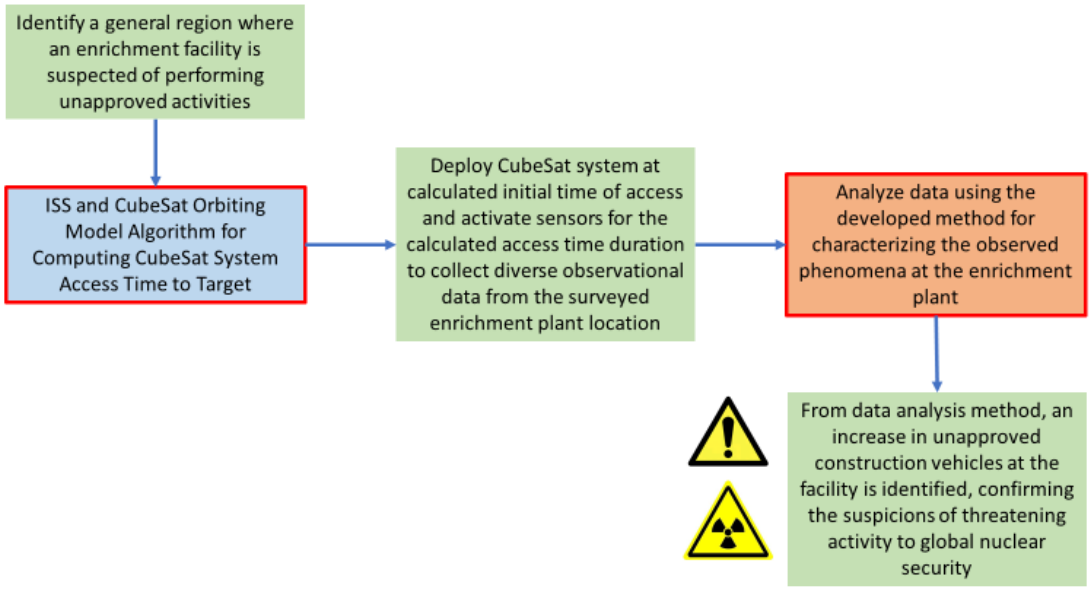


Figure 21. The data flow for observing and identifying suspicious activity at an enrichment plant.

3.4. Conclusion

This section explored different architecture and deployment scenarios for the CubeSat system. The system was determined to be comprised of a constellation of multiple satellites instead of one. The main reason for the decision was driven by a constellation's higher characterization accuracy due to a higher number of sensors in the system. Since a constellation was chosen for the system, different communications configurations were explored. Intersatellite communication using a satellite relay network, like Globalstar, combined with a network of ground stations, like KSAT was determined as the optimal communications option. Also, a satellite system architecture featuring a single CubeSat as the major computational unit for processing most of the data collected by the sensors was

chosen. The last topic of the section introduced a mathematical algorithm which utilizes GMAT for determining CubeSat deployment times.

4. SENSOR SELECTION

Sections 2 and 3 discussed in detail the capability of a CubeSat system to support a surveillance platform and its configuration. The sections determined that a constellation of CubeSats deployed from the ISS with adequate power, communications, and other hardware capabilities proves to be a feasible surveillance system with lower costs and higher simplicity with respect to conventional satellites. This section joins the CubeSat system developed thus far with the considerations about the phenomena of interest for surveillance by selecting sensors for the system. As was seen in the design process presented in Section 1, the sensor selection for the system influences the creation of the surrogate data set used to develop the characterization methodology in the next section. This section first analyzes the surveillance requirements for observing signatures from the phenomena of interest to define the required resolutions for sensors onboard the CubeSat surveillance system. Then, the different types of sensors available for meeting the surveillance requirements are studied and recommendations are made. Finally, different satellite system requirements for the integration of sensors are considered.

4.1. Sensor Introduction

With the advent of advanced aerial and spatial platforms, higher resolution sensors, and data processing capabilities, remote sensing system are becoming increasingly popular for collecting data on the Earth and its processes. Remote sensing platforms allow the user to obtain critical data from vantage points not readily accessible to humans. The first instances of remote sensing occurred in the mid-1800s and early 1900s when balloons and, subsequently, airplanes were used for surveying land in commercial and military

applications [87]. Although airplanes and, most recently, drones still present an effective platform for remote sensing systems with very high resolution, satellites offer reliable, quick, and repeated global coverage of even the most remote locations around the globe, often times without being detected themselves by their targets on the ground. A satellite-based remote sensing system allows scientists and governments to collect data never thought possible before. Currently, typical civilian uses for satellite systems include: monitoring biomass energy and foliage inventory, determination of solar energy absorption on the planet's surface, monitoring of snow cover, monitoring of weather patterns and natural disasters, pollution detection, monitoring time changes in water systems, monitoring urban growth, and many other applications. Recently, NASA and the National Oceanic and Atmospheric Administration (NOAA) have launched two satellites since 2016 and plan on launching two more by 2024 as part of their Geostationary Operational Environment Satellites (GOES) program to provide advanced imagery and high-accuracy measurements of meteorological events [88]. The NASA GOES satellites will accomplish their remote sensing goals through the use of an advanced baseline imager, an extreme ultraviolet x-ray irradiance sensor, a geostationary lightning mapper, a magnetometer, a solar ultraviolet imager, and a suite of in-situ sensors for radiation flux monitoring in space [89]. As for governmental or military operations, satellite-based remote sensing platforms provide important data used for monitoring, targeting, strategic planning, deployment, and threat assessment [90]. Although information over specific remote sensing capabilities or technology on military satellites is not readily available, general sensor technologies used for accomplishing the goals of remote sensing military

satellites include: optical imagers that gather visible light, multispectral imagers, hyperspectral imagers, and radio or microwave scanners [90].

A remote monitoring system focused on effectively characterizing events pertaining to nuclear fuel cycle surveillance, as the one developed in this system, benefits most from a satellite platform because of all the benefits over aerial platforms previously mentioned. The CubeSat-based surveillance system introduced in this thesis provides a simpler and cost-effective platform when compared to conventional satellites. The biggest disadvantage for remote sensing on a CubeSat platform, however, is the loss of data resolution when compared to larger satellites. For example, the current highest spatial resolution CubeSat panchromatic/multispectral imagers reach a resolution of 1 m, while larger satellites can feature up to a 30-40 cm resolution [68,91,106]. It is worth noting that the 30-40 cm resolution is the highest available resolution for only commercially available images. The difference in resolutions stems mainly from the restraints on aperture sizes for imagers within a CubeSat structure.

The following subsections explore the sensor options for the CubeSat surveillance system by considering the sensor requirements for observing the phenomena of interest based on their signatures and currently available technology. It is worth mentioning that remotely sensing radiation was a very attractive proposition for a surveillance system focused on monitoring nuclear facilities. It was decided that, unless any nuclear weapons detonations occurred, radiation signatures are not likely to be detected from LEO. Due to the distance of the sensor to the radiation source at a facility and the density of the atmosphere, any trace of radiation detected would be indiscernible from the background

radiation at the CubeSats' altitude. Eq. 4-1 gives the attenuation equation used during the determination. At an average ISS altitude of 400 km, only 0.028% of a 20 MeV gamma source's intensity would reach a radiation detector.

$$I = I_0 e^{-\mu x} \tag{4-1}$$

Where,

$I =$ The intensity of photons at distance x across a medium
$I_0 =$ The initial intensity of the photons
$\mu =$ The linear attenuation coefficient of the medium
$x =$ The distance the photons traveled in the medium

4.2. Surveillance Requirements

For the CubeSat remote monitoring system to correctly characterize the phenomena occurring on the planet's surface, sufficient data must be collected with the system sensors to adequately describe the phenomena. When the phenomena of interest were introduced in Section 1, several parameters and signatures describing each phenomenon were given. For example, the dimensions of typical commercial trucks were given, as well as their engine temperatures and emissions. Allowing the CubeSat system to collect diverse kinds of data was the driving factor in choosing a constellation system architecture, as was seen in Section 3. Although some phenomena, like a commercial truck, can be characterized by only one kind of signature, like a commercial truck's dimensions, the characterization false positive and false negative rates decrease if data from other signatures is also collected, like a commercial truck's emissions. Therefore, the goal of down selecting sensors for the surveillance system involves collecting as much information as possible from the phenomena. Two main considerations were made in the sensor selection process in this section; signature requirements and system requirements.

It is important to note that this section only identifies the types of sensors needed that best suite the remote monitoring goals of the CubeSat system. Specific sensor systems were not chosen.

Methods and sensors for remote sensing, especially for Earth observation, can be divided into two categories; Passive and Active imagery [92]. In principle, the difference between the two methods involves how sensors in either category collect their data. In Passive imagery, sensors collect emissions and signatures from the subject of observation. The signals the sensor collects can either be directly produced from the subject (like thermal infrared radiation) or be reflected off the subject (like visible light) [92]. Active imagery sensors on the other hand rely on sending and receiving signals in the electromagnetic spectrum to collect data on the object of interest, like radar. Passive sensors usually operate in the visible (0.39 – 0.70 μm) and infrared (0.70 – 14 μm) spectra, and typical imagery types include; panchromatic, multi-spectral, and hyper-spectral [92]. Active sensors usually operate in the radio wave (1 cm – 11 m) spectrum, and typical imagery types include; synthetic aperture radar (SAR), light detection and ranging (LIDAR), radar altimetry, and radar scatterometry [92]. Although both sensor types provide suitable options for the CubeSat surveillance system, the analysis for the phenomena of interest signatures and sensor recommendations are mainly focused on passive imagery.

As mentioned, passive sensors within a remote monitoring system are limited to the signatures emanating from the subject of observation which can reach the system. For example, if a scientist were to measure the temperature of an exothermic chemical

reaction, they would place a thermometer within the solution undergoing the reaction since the scientist has direct access to the subject of observation. A remote monitoring system on the other side of the laboratory does not have that kind of access, so the temperature of the chemical reaction cannot be measured via a thermometer. The remote monitoring system is, therefore, forced to measure the temperature of the chemical reaction through other means. Since the solution undergoing the chemical reaction is producing heat, it is producing infrared radiation. Infrared radiation, or infrared light, is part of the electromagnetic spectrum, having wavelengths just larger than visible light [93]. The emittance of infrared radiation by objects is what humans perceive as heat. By detecting the infrared signatures from the chemical reaction, the remote sensing system at the other side of the lab can determine the temperature of the reaction. For detecting infrared radiation a few meters away from the subject of observation, a typical thermal imaging camera can be used to capture the chemical reaction's temperature [94]. Like with the chemical reaction in the previous example, the phenomena of interest for the CubeSat remote monitoring system also produce different kinds of signatures detectible by a satellite in LEO. When analyzing the capabilities of a remote sensing system to detect a certain phenomenon, especially from hundreds of kilometers away, the phenomenon's signature must be separated into spatial, spectral, and temporal resolutions. Defining the needed resolutions to detect signatures of each phenomenon parameter is the first step in selecting a sensor to measure such parameter. Table 16 summarizes the different types of signature resolutions the remote sensors can detect for each phenomenon of interest parameter. During the parameter signature identification process, the resolutions were

defined in the following order; spectral, spatial, and finally, temporal. The spectral resolutions needed for each signature were analyzed first because it is the biggest factor in determining the type of sensor needed for that parameter's signature. For example, an optical imager operating only in the visible light spectrum is not able to collect data reflectance data in the infrared spectrum for gases in the atmosphere. It is important to note that the "spectral resolutions" recorded in Table 16 actually represent spectral "regions" where the parameters can be observed, like the visible light range. In the remote monitoring community, the spectral resolution of a sensor refers to the distance between the bands in the data. Therefore, a sensor operating in the visible light spectrum (0.39 – 0.70 μm) may contain only three separate bands (red, green, and blue). While the spectral regions determine the type of sensor technology, the spatial resolution required for the parameters determines the quality and technology level of a specific sensor, and the temporal resolution determines the time of observation with the sensor. It is important to note that the required spatial resolutions were determined by the physical sizes of the phenomena as well as typical resolution regions for previous satellite imagery. Typical resolution regions are defined as high to very high (30 cm – 5 cm per pixel), medium (10 – 30 m per pixel), and low (over 60 cm per pixel).

Table 16. The spectral, spatial, and temporal resolution needed for characterizing a phenomenon's parameters.

Phenomenon	Parameter	Signature Resolution		
		Spectral (Region)	Spatial	Temporal
Automobiles and Airplanes	Length, Width, Height	Visible and Infrared Light bands	1-1.5 m	Single Sample
	Speed	Visible and Infrared Light bands	1-1.5 m	Multiple Samples
	Temperatures	Infrared Light bands	1-1.5 m	Single Sample
	Gas Emissions	Infrared Light (Reflectance) bands	10-30 m	Continuous Sampling
Facilities and Emergencies	Length, Width, Height	Visible and Infrared Light bands	10-30 m	Single Sample
	Temperatures	Infrared Light bands	10-30 m	Single Sample
	Gas Emissions	Infrared Light (Reflectance) bands	10-30 m	Continuous Sampling
	Aerosol Index	Ultra-Violet Light bands	10-30 m	Continuous Sampling
Construction and Mining	Length, Width, Height	Visible and Infrared Light bands	1-2 m	Single Sample
	Speed	Visible and Infrared Light bands	1-2 m	Multiple Samples
	Temperatures	Infrared Light bands	1-2 m	Single Sample
	Gas Emissions	Infrared Light (Reflectance) bands	10-30 m	Continuous Sampling
	Footprint	Visible and Infrared Light bands	10-30 m	Single Sample

For observing the dimensions of the different phenomena from LEO, a sensor can utilize both the visible light and infrared spectral regions. Distinct features of an object, like length, width, and height, can be identified in both regions by analyzing a difference in wavelengths received from the object and its environment. In the visible range (0.39 – 0.70 μm), the difference in wavelength translates to differences in colors. In the infrared range (0.70 – 14 μm), the difference in wavelength translates to a difference in temperatures or emissivity. While data in the visible light range may provide more detail about an event given appropriate conditions, data in the infrared range can provide adequate data through cloud cover or at night [95]. Therefore, the CubeSat surveillance system will benefit from having sensors operating in both spectra for the identification of object dimensions. As for spatial resolutions when observing dimensions, automobiles and airplanes need a resolution of 1-1.5 m since the smallest value for a dimension in that phenomenon category is 1.41 m, as can be seen in Section 1. Sensors with resolutions lower than the values of the parameters prevent the parameters from being recognized, causing the subject to not show up in the data collected. For the vehicles in the construction and mining events of interest, a similar resolution is needed since the smallest value for a dimension is 1.62 m. As for the length, width, and height of the objects of interest in the facilities and emergencies group, at least a medium resolution level of 10-30 m is needed. A higher spatial resolution, like for detecting vehicles, can still be used but is not needed due to the size of buildings at facilities. As for needed temporal resolutions for detecting height, length, and width, only one instantaneous reading is necessary. As will be discussed later in this section, most sensors considered for the CubeSat system capture

data instantaneously. Detecting speed, however, requires a temporal resolution of multiple measurements during a single observation. These multiple measurements can be taken by the same sensor at different times or by two identical sensors from the same spot at different times. A constellation of CubeSats can account for needed temporal resolutions in both regards. The speed of an object can be calculated by comparing the difference in position of an object between to identical readings at different points in time. The spectral and spatial resolutions for detecting speed are the same as for the dimension parameter.

For detecting temperatures in all three phenomena categories, a spectral region within the infrared radiation wavelengths (0.70 – 14 μm) is needed. All objects emit energy in the form of infrared radiation which describes their temperature [96]. The hotter a material, the shorter the wavelength of the electromagnetic radiation it produces. When a material starts reaching high enough temperatures, this electromagnetic radiation enters the visible light spectrum, like with fires. By detecting the wavelengths of the infrared radiation of an object, its temperature can be determined. Spatial resolutions for temperature readings vary between the different categories as they did for dimension readings. For all three categories, a spatial resolution equal to the one needed for detecting their individual dimensions can be used to determine the specific temperatures of the objects within their dimensions, as mentioned earlier. It is worth noting that the characterization of phenomena based on their temperature readings is not necessarily restricted to an object's dimensions. For example, due to radiative heat transfer and engine emissions, automobiles and airplanes create a temperature "footprint" which is larger than the vehicle [97]. A larger area for detecting temperature "footprints" could translate to

lower spatial resolutions, but it was determined that utilizing lower resolutions from what was already specified would cause too much detail to be lost about the phenomena. Temporally, typical sensors that operate in the required spectral and spatial regions for temperature readings make instantaneous observations.

For characterizing gas emissions, the most common remote sensing technique is spectroscopy. Sensors measure the wavelengths absorbed or emitted by a sample, generally in the long-wave infrared spectrum (8 – 14 μm), to characterize the chemical makeup of a sample [98]. A specific chemical compound will have a unique spectral signature of absorbed and reflected wavelengths that a sensor can use to identify it. The gas emissions from all three phenomenon categories can be identified by their unique reflectance bands in the infrared light spectrum. Spatially, a medium range resolution of 10-30 m should be adequate for characterizing the gas emissions in all phenomena categories. Spatial resolutions higher than that are generally not worth the additional sensor costs since gases released to the atmosphere do not tend to stay constrained to the area in which they were created [99]. As for temporal resolutions, typical sensors used for spectroscopy in this fashion take continuous readings throughout the observational period to record the highest amounts of data possible [98].

For calculating the aerosol index of events under the infrastructural emergencies category, measurements taken in the near-ultra-violet (UV) spectrum (0.27 – 0.39 μm) are best suited. The aerosol index refers to the number of particles present in the atmosphere at a given location and can be quantified by measuring the amount of UV radiation they absorb [100]. If more UV light is absorbed in a section of the atmosphere, the more non-

air particles are present. Although the presence of aerosols can be measured in the UV, visible, and infrared spectra, the typical aerosol index calculations introduced in Section 1 are conducted with data obtained in the UV spectrum. Just like with characterizing gas emissions, a spatial resolution of 10-30 m is adequate for measuring aerosol indices. A higher spatial resolution would be overkill, and a lower resolution is not enough to adequately characterize the phenomena. For temporal resolutions, typical sensors in the UV or infrared range that measure atmospheric quality take continuous readings throughout the observation time of a phenomenon [98,100,101].

Finally, the last parameter to be defined was the footprints for mining and construction activities of interest. To identify footprints, sensors operating in the same spectral regions as for identifying the dimension parameters of the other phenomena are suitable. Considering the typical sizes for mine footprints, a spatial resolution of 10-30 m is enough to identify the phenomena. As for temporal resolutions, it has been mentioned that sensors in the visible and infrared spectrum can capture data immediately.

4.3. Sensor Recommendations

Given the signatures and resolutions presented in Table 16, considerations and recommendations for the types of sensors onboard the CubeSat surveillance system can be made. As was mentioned, the spectral regions required for observing the phenomena of interest from a passive imaging perspective had the biggest influence in the sensor selection. Table 17 summarizes the sensor recommendations for identifying each parameter for the phenomena of interest.

Table 17. The sensor type recommendations for observing each parameter.

Phenomenon	Parameter	Sensor Recommendation
Automobiles and Airplanes	Length, Width, Height	Panchromatic and Multispectral
	Speed	Panchromatic and Multispectral
	Temperatures	Multispectral
	Gas Emissions	Hyperspectral
Facilities and Emergencies	Length, Width, Height	Panchromatic and Multispectral
	Temperatures	Multispectral
	Gas Emissions	Hyperspectral
	Aerosol Index	Multispectral
Construction and Mining	Length, Width, Height	Panchromatic and Multispectral
	Speed	Panchromatic and Multispectral
	Temperatures	Multispectral
	Gas Emissions	Hyperspectral
	Footprint	Panchromatic and Multispectral

For observing the length, width, and height of all three categories along with the footprints in the mining and construction events of interest category, panchromatic and multispectral sensors are best suited. Panchromatic imagers typically produce the highest spatial resolutions and can operate in either the visible light spectrum or thermal infrared spectrum (10 – 12 μm) [92]. Although sensors operating in the visible range may produce more detailed images, sensors in the thermal infrared range can be produce images at night. A surveillance system featuring sensors collecting data in both spectra makes it less dependent on environmental conditions and produces more accurate results. Panchromatic data is produced through one band that spans the region of interest in the electromagnetic spectrum, combining information from all the wavelengths within the region instead of producing separate spectra [102]. The information collected per pixel represents the intensity of light received, producing images within the panchromatic band as grayscale [103]. Multispectral imagers, on the other hand, collect data across more spectral bands

within regions of the electromagnetic spectrum [103]. Separating the signals received into various bands allows the sensor to collect more information on the wavelengths of the light received. For example, colored pictures are created with multispectral sensors collecting data across a few bands. Common bands for multispectral imagers best suited for observing details in dimensions are the blue, green, red, and/or near-infrared bands (0.45 – 0.9 μm). Since multispectral sensors divide the energy of light they receive across multiple bands, each individual band receives less energy when compared to the single panchromatic band. Due to the reduced energy received, multispectral sensors need to sample a larger area to collect the minimum amount of energy needed for identifying the differences in brightness for each band wavelength, causing them to generally have lower spatial resolutions than panchromatic sensors [103]. In order to collect the highest amount of accuracy spatially and spectrally when observing phenomena of interest, panchromatic and multispectral images can be combined to utilize the benefits from both types of data [92]. This process of data fusion is referred to as pansharpened data. Historically, pansharpened images are best suited for object-based image analysis, which is needed for identifying the dimensions of the phenomena of interest [103]. The CubeSat system will best meet its surveillance goals for the phenomena of interest using pansharpened data. Therefore, the CubeSat constellation will feature at least one panchromatic sensor in the visible spectrum and one multispectral sensor operating in the blue, green, red, and near-infrared bands. Featuring four bands in the multispectral sensor will provide greater diversity in data collected, increasing the characterization accuracy. For example, data collected in the near-infrared band of a multispectral sensors can aid in distinguishing

man-made structure from vegetation by displaying the difference in their emitted infrared radiation as distinct colors in the images [92]. Easily identifying vegetation in a scene through a difference in color can increase the characterization of phenomena located in regions with dense vegetation. Also, the CubeSat system surveillance versatility will increase by featuring a panchromatic sensor operating in the infrared region to identify phenomena at night.

Detecting temperatures from satellites can be accomplished with multispectral sensors. Like the multispectral imagers previously mentioned for observing phenomena dimensions, a multispectral imager used for temperature measurements collects data on several bands within a region of the electromagnetic spectrum. The region of operation for multispectral sensors suited for temperature measurements is typically in the infrared spectrum (8 – 14 μm) [104]. Their functionality is the same as for multispectral imagers in the visible and near-infrared regions, but it is done in the long-wave infrared region. Once data is collected in each band, images are produced where different colors indicate different temperatures in the observed scene. As was previously mentioned, panchromatic sensors can also operate in the infrared region and produce a higher spatial resolution than multispectral sensors. Since they only have a single band, thermal panchromatic sensors cannot produce the level of spectral detail necessary for identifying distinct temperatures. They only observe a “change” in temperatures between object. A multispectral sensor, on the other hand, can observe greater detail between the temperature values within a scene due to its greater number of bands. Essentially, the difference in detail between panchromatic and multispectral sensors for thermal imaging is the same as the difference

between grayscale and color images in the visible light region. For the purposes of observing the temperatures of phenomena from the CubeSat surveillance system, the level of detail produced by thermal multispectral sensor is best suited.

As was mentioned previously, traces of gas emissions in the atmosphere can be identified through spectroscopy of the absorbed and reflected wavelengths from a sample. Hyperspectral sensors can accomplish this task by featuring hundreds of bands in the infrared spectrum [98]. Each band within a hyperspectral sensor collects data in the same way the previously mentioned sensors due, but while a multispectral sensor may have a maximum of 15 bands, hyperspectral sensors can have more than a hundred [103]. The unique spectral signatures of chemical compounds are identified by detecting the intensities of infrared light received at each of the narrow bands of a hyperspectral sensor. Data collected from hyperspectral sensors can be thought of as a data "cube" having two dimensions being spatial and the third being spectral. As a hyperspectral sensor on an aerial or spaceborne vehicle passes over a target, data is usually collected in a "push broom" fashion, which means that measurements need to be taken continuously to form an image instead of taking an instantaneous picture [98]. In each individual measurement, the sensor collects spectral data within its instantaneous field of view (IFOV). The IFOV has a specific spatial resolution in the x and y dimensions, and it creates a single pixel in the hyperspectral image [98]. As the sensor makes continuous measurements, its IFOV sweeps through its swath path to create a full image out of each pixel collected. The CubeSat surveillance system will be able to identify the presence of gas emissions through use of a hyperspectral sensor.

As for measuring the presence of aerosols in the atmosphere and calculating an aerosol index, a multispectral sensor is suitable. Although the identification of gas emissions with hyperspectral sensors can also describe the presence of aerosols in the atmosphere, too much unnecessary data is collected if the parameter of interest is just simply the aerosol index. A multispectral sensor, on the other hand, only leads a few spectral bands to measure the presence and absence of aerosols (the aerosol index) when observing a phenomenon since the specific identity of the aerosols is not needed. Although aerosol indices have been historically calculated using data from multispectral sensors operating in the UV spectrum, atmospheric quality can also be measured within the infrared spectrum given adequate data postprocessing [100,105].

After determining the types of sensors needed for each parameter, the technology readiness level (TRL) for the sensors at the required spatial resolutions listed in Table 17 was analyzed. Table 18 summarizes current sensor capabilities of meeting those required spatial resolutions for a CubeSat platform [68,74,106,107,108,109,110,111,112,113,114]. The table represents the results of an extensive literature review on CubeSat heritages for each sensor type and at the needed spatial resolutions. Although all recommended sensor types have been previously operated on CubeSat platforms, their current TRL for the CubeSat surveillance system in this thesis varies with sensor type. There currently exists panchromatic and multispectral sensor technology in the visible and near-infrared spectrum capable of detecting the phenomena object dimensions, speeds, and footprints [106]. As for multispectral and hyperspectral sensor technology in the infrared or UV region for observing temperatures, gas emissions, and aerosol indices, future development

is needed to meet the spatial requirements for the phenomena of interest. As was mentioned in earlier sections, any sensor development efforts for the completion of a final CubeSat system ready for deployment is outside the scope of this work.

Table 18. Sensor type recommendations and CubeSat heritage at required spatial resolutions.

Phenomena	Parameter	Sensor Type	CubeSat Heritage	CubeSat Heritage at Spatial Resolution	Future Sensor Development Needed?
Automobiles and Airplanes	Length, Width, Height	Panchromatic/ Multispectral	Yes	Yes	No
	Speed	Panchromatic/ Multispectral	Yes	Yes	No
	Temperatures	Multispectral	Yes	No	Yes
	Gas Emissions	Hyperspectral	Yes	No	Yes
Facilities and Infrastructural Emergencies	Length, Width, Height	Panchromatic/ Multispectral	Yes	Yes	No
	Temperatures	Multispectral	Yes	No	Yes
	Gas Emissions	Hyperspectral	Yes	No	Yes
	Aerosol Index	Multispectral	Yes	No	Yes
Construction and Mining Events	Length, Width, Height	Panchromatic/ Multispectral	Yes	Yes	No
	Speed	Panchromatic/ Multispectral	Yes	Yes	No
	Temperatures	Multispectral	Yes	No	Yes
	Gas Emissions	Hyperspectral	Yes	No	Yes
	Footprint	Panchromatic/ Multispectral	Yes	Yes	No

As was mentioned, the previous analysis on parameter signatures and sensor recommendations assume passive imagery methods only. However, active imagery sensors, like synthetic aperture radar (SAR) and light detection and ranging (LIDAR), can also be used for observing the phenomena of interest [92]. Active imagery sensors send and receive signals in the electromagnetic spectrum to collect data on the object of interest. All of the parameter types listed for the phenomena can be observed through either SAR or LIDAR techniques [92,115,117,118,119]. However, passive sensors were chosen as the main sensor recommendations because of their increased TRL and hardware simplicity when compared to active sensors. Traditionally, CubeSat missions for Earth observation have stayed clear of active sensors due to their large size, weight, and power requirements [119]. In recent years, multiple efforts have worked to dispel that notion by developing adequate SAR, LIDAR, and other radar systems for CubeSat platforms, with some currently in orbit [117,119,120]. Out of all the active sensors under development for CubeSats, none are currently capable of meeting spatial requirements for the phenomena of interest. Also, active sensors require additional hardware installations, like antennas, that may increase the time and complexity of the CubeSat fabrication process when compared to passive sensors. If sufficient sensor development efforts are completed in the future, SARs and LIDARs present viable sensor options for the CubeSat surveillance system.

4.4. System Requirements

When making final sensor selections for a final CubeSat surveillance system given the sensor recommendations in the previous section, the system requirements for sensor

integration with the CubeSat must be analyzed. The biggest factors affecting the feasibility of installing a sensor on a CubeSat are size and power. As was seen in Table 18, all the sensor types recommended for the surveillance system have previous experience on CubeSat systems, meaning that size and power requirements have been met in the past. As future sensor developments are made to increase spatial resolutions, it is important that the sensors can still fit within a CubeSat form factor while still providing enough space for the additional satellite systems. As mentioned in Section 2, the most common CubeSat mission sizes are 3U and 6U, and they can have a maximum size of 12U. As an example, a 3U CubeSat can house a sensor no larger than a 2U size.

As for power, Section 2 mentioned that a typical CubeSat power supply can offer up to 80 Wh. This means that the sensor and other satellite components have 80 W of power available for an hour before the batteries need to be recharged. Since some components require a constant stream of power and others may require high power consumptions for short periods of time, an adequate duty cycle must be created for the power supply system. A duty cycle coordinates the power consumption of various components with the power generation from the solar panels and battery capacity to ensure there is always enough power for the satellite system. In a typical CubeSat mission, the components with highest power consumption are the sensor and transmitter/receiver system and should therefore not operate simultaneously to ensure enough power is available to the operational component of the two and to the rest of the satellite. Assuming an average CubeSat power consumption of 15 W, excluding the sensor and transmitter/receiver system, there are up to 65 Wh of energy available for use [108]. If the

longest possible duration for an access time, as mentioned in Section 2, is around 7 min, then a sensor would ideally have up to 557 W available during an observation. However, the more power a sensor uses during its operation, the longer it will take for the battery to recharge with the solar panels. If the battery cannot recharge in time before a subsequent access time, there will not be enough available power for the sensor to make another observation. Therefore, adequate duty cycle planning is necessary for operations. Figure 22 shows an example duty cycle graph with the power generation, consumption, and capacity for a CubeSat for a 24-hour period. The CubeSat has two access times to a location within the 24-hour period where it turns on its detector, like in the example in Section 2. For the sake of the example, it was assumed that each access time lasted 10 min, which is around the maximum access time a CubeSat can have to a target given its orbit. Also, the CubeSat made multiple data transmission in 10- and 5-min increments. The power generation comes from solar panels producing 40 W whenever they are in the sun, and the power consumption assumes an average 15 W for the CubeSat bus at all times, an additional 14 W for the transmitter/receiver system, and 80 W for the sensor [77,82,108]. As can be seen, the CubeSat in the example is adequately able to house an 80 W sensor. The plot was created by first defining the generation and consumption functions and then calculating the battery capacity. All three functions were defined as a function of time with increments of 5 min throughout the 24 hours. For the power generation function, it was defined at 40 W between $t = 0\ min$ and $t = 45\ min$ and then at 0 W between $t = 45\ min$ and $t = 90\ min$. The 45-minute step function value increments were alternated until the time reached 24 hours. For the power consumption function, its values were

initially set at 15 W for all t. Then, the power values for the sensor and transmitter/receiver system were added to either 5-min or 10-min t increments throughout the 24-hour period. The battery capacity function was then calculated at each individual time step. The battery capacity at $t = 0$ was set to 80 Wh, and then Eq. 4-2 was used to calculate the power capacity at each subsequent time step. Since the maximum battery capacity is only 80 Wh, if the calculated capacity with Eq. 4-2 ever exceeded that value for a time step, the calculated value was capped at 80 Wh.

$$Capacity_i = \left[\left(\frac{Capacity_{i-1}}{\Delta t}\right) + Generation_i - Consumption_i\right] * \Delta t \qquad (4\text{-}2)$$

Where,

$Capacity_i$ = Battery capacity calculated at time step i in Wh
$Generation_i$ = Power generated at time step i in W
$Consumption_i$ = Power consumed at time step i in W
Δt = Time step of 0.083333 hrs (5 min)

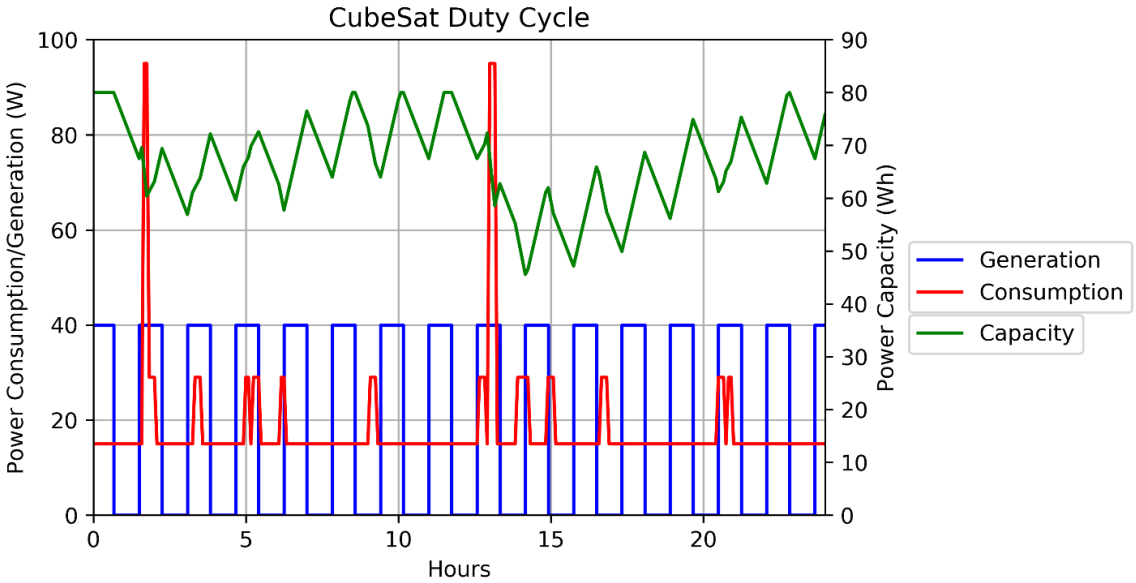


Figure 22. An example duty cycle for a CubeSat with an 80 W sensor and two access times in 24 hours.

Once final sensor selections have been completed, the final considerations for system performance are the sensors' angles of elevation for observation. Although the angles of elevation do not affect the capability for sensor integration into a CubeSat, they affect CubeSat operations once in orbit. As was seen in Sections 2 and 3, a larger angle of elevation creates a smaller viewing angle for a sensor and, in turn, creates a shorter access time to a target. Selecting sensors with smaller angles of elevation (larger viewing angles) allots the surveillance system more time to collect data and are recommended.

4.5. Conclusion

The section first explored the capabilities of remote monitoring and considered the surveillance requirements for the phenomena of interest. Defining the signatures for the phenomena determines the type of sensors needed for the CubeSat surveillance system. For each phenomenon of interest, their signature resolutions in the spectral, spatial, and temporal dimensions were established. Then, the different sensor requirements necessary for monitoring the signatures were explored. The various kinds of sensors and their capabilities were defined. It was determined that the CubeSat surveillance system can accomplish its observation goals by featuring a combination of panchromatic and multispectral sensors operating in the visible and near-infrared spectrum, multispectral sensors operating in the infrared spectrum, hyperspectral sensors operating in the infrared spectrum, and multispectral sensors operating in the ultraviolet spectrum. The current capabilities and CubeSat heritage for the recommended sensors were explored, and it was observed that future sensor development is needed to reach the necessary resolution and

size requirements. The section concludes with an analysis of CubeSat satellite architecture capabilities of housing sensors.

5. SURROGATE DATASET

While the previous three sections analyzed the physical capabilities of CubeSats and all their components as a surveillance system, the focus is now shifted towards the development of the characterization methodology on board the satellites. As has been previously alluded to, the methodology developed in the rest of this thesis is based on machine learning techniques, specifically deep learning. Like with any deep learning task, an adequate dataset in terms of quality and size is required to develop an accurate model. The dataset created in this section for training the deep learning model for characterization aims at representing the data collected from the different sensors on board the surveillance system at a proof of concept level. Rather than training the methodology on real satellite imagery, the dataset is a simplified representation. This approach is taken because of a lack of actual sensor data from a CubeSat platform, and to allow the methodology developed to serve as a proof of concept for deep learning models that can adapt to other types of data. Creating the characterization methodology on a dataset too specific to certain scenarios can limit the methodologies adaptability to different kinds of data. Also, the surrogate data set assumes that the raw data collected from the system's sensors is sufficiently post-processed so that only the relevant information is fed to the characterization methodology. There are multiple machine learning and data fusion techniques employed at improving the data produced by sensors, like interpreting hyperspectral data as an image or reducing the signal-to-noise ratio (SNR). Such data processing techniques are outside the scope of the characterization methodology. This

section first introduces deep learning and its applications. Then, the methodology for creating the surrogate dataset and results are discussed.

5.1. Introduction to Machine and Deep Learning

In recent years, machine learning (ML) and artificial intelligence (AI) have been the dominating buzzwords for most engineering industries. Since the 1950s, the field of AI has expanded from solving well-defined problems using logic, like playing Go and chess, to more complex and open-ended problems, such as language processing, computer vision, data fusion, and much more [121]. Machine learning is a subset of AI and is comprised of techniques that "learn" representations from data. ML algorithms use large datasets of known examples of inputs and outputs to produce, or "learn", statistical models for the correlations between the inputs and their outputs. In other words, ML models can "learn" to predict accurate outputs given new inputs after enough exposure to examples of similar data. A ML algorithm encodes the data it was given into representations that are better interpreted for finding the correlations between inputs and outputs. For example, given enough pictures of dogs and cats with the correct labels, a model can predict whether a new picture is that of a dog or a cat. The downfall of ML, however, is that it is heavily reliant on the data used for training the model. In the dogs and cats example, the model cannot identify a picture of an elephant since it was only exposed to pictures of dogs and cats. Therefore, the dataset created in this section must represent phenomena as captured by "sensors" in a simplified way accurately enough for the characterization algorithm to effectively characterize the data.

When attempting to create a dataset, it is important to consider two separate branches of machine learning, supervised learning and unsupervised learning. Supervised learning involves an algorithm learning representations on data which include their known targets [121]. In other words, examples of data and their solutions are used to predict solutions for new data. Typical applications for supervised learning include classification and regression. On the other hand, unsupervised learning involves an algorithm learning representations from data which do not have known targets or solutions [121]. Unsupervised learning can aid in understanding data through visualization, compression, or noise reduction. Typical applications for unsupervised learning include dimensionality reduction, clustering, and anomaly detection. Since the characterization methodology for the CubeSat surveillance system aims at identifying phenomena from sensor images, it can be defined as image classification, which falls under supervised learning. Each piece of data in the surrogate dataset must also include a label describing it. For example, a surrogate "image" of an automobile will include a label of "automobile".

5.1.1. Deep Learning Basics

In the same way ML is a subset of AI, deep learning is a subset of ML. While other ML techniques only learn single "layers" of representations of data, deep learning techniques use multiple "layers" for learning representations from the data [121]. In other words, these successive layers learn representations from the previous representations of the data. The word "deep" in deep learning refers to "depths" of models containing multiple successive layers [121]. These deep models are referred to as "neural networks". Each layer in the network contains multiple perceptrons, or neurons, as can be seen in

Figure 23. A neuron takes in one or multiple inputs *x* (this could be the pixel values for the picture of a dog) and produces an output *y* (this could be the label of "dog" for the picture) through Eq. 5-1. The weights *w* and biases *b* in Figure 23 and Eq. 5-1 are randomly initiated and are continuously adjusted as more data is passed through the neuron until it "learns" the correct values of *w* and *b* that produce a *y* that matches the *y* from the data given a metric for weighing performance, like the mean squared error or accuracy.

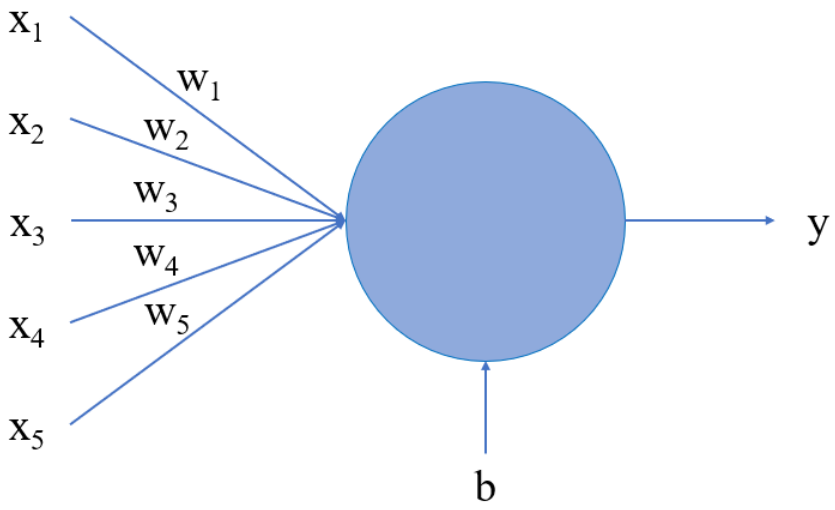


Figure 23. A visual representation of a neuron.

$$y = \sum_{i=1}^{N} w_i x_i + b \tag{5-1}$$

Where,

x = an input value
w = a weight value
b = a bias value
y = an output value

For a complete layer within a network, the neurons take in the input as a tensor **x**, use tensors for weights **w** and biases **b**, and output a tensor **y**. Eq. 5-1 for a single neuron is then changed to Eq. 5-2 for a layer. Eq. 5-2 represents the linear combination of

parameters executed within a layer, but each layer also performs a non-linear combination known as the activation function [122]. Instead of just outputting the solution **y** of the linear combination, each layer in the network outputs the solution of the activation function, f(**y**). A common, well-performing activation function (and the one used for the characterization network in the next section) is the rectified linear unit function, or ReLU(**y**), and it is described with Eq. 5-3 and Figure 24 [122]. With each layer of neurons in the network the output f(**y**) then becomes the input **x** for the next layer. A visual representation of a simple neural network can be seen in Figure 25.

$$\mathbf{y} = \mathbf{wx} + \mathbf{b} \tag{5-2}$$

$$ReLU(\mathbf{y}) = max\{0, \mathbf{y}\} \tag{5-3}$$

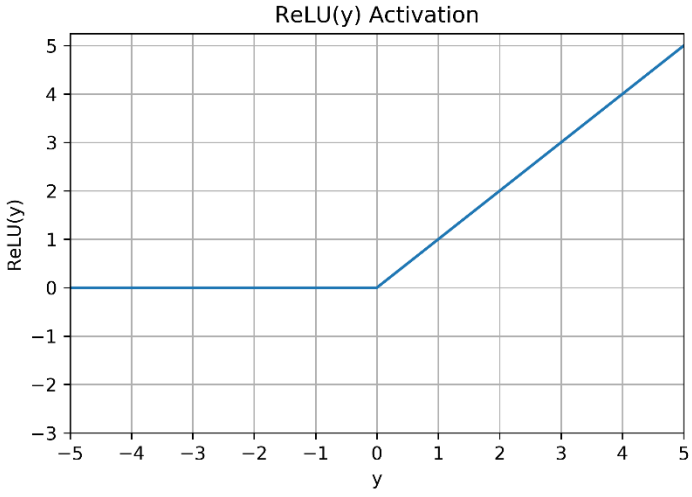


Figure 24. A plot of the ReLU(y) activation function given y.

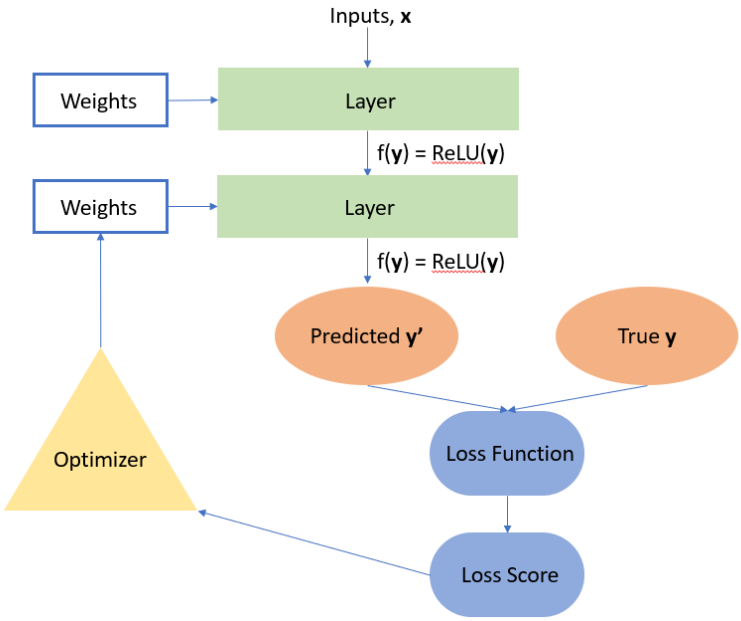


Figure 25. The architecture of a simple neural network.

As was mentioned and as can be seen in Figure 25, once a prediction, **y**', is made on the data, it is compared to the true **y** to measure the performance of the network through the loss function. There are many different types of loss functions employed in neural networks, but a popular choice is the mean squared error, as described in Eq. 5-4 [122]. The type of loss function depends on the object of the neural network, however. For example, a network trained for regression tasks might use the mean squared error as a loss function, but a network trained for classification might use binary cross-entropy. The loss function used for the characterization algorithm is described in the next section. Once the loss function is used to calculate the loss score for a network, an optimizer function is used to update the network parameters (weights and biases) in order to reduce the loss score in the next iteration. This updating of the network parameters with every iteration is the

essence of the "learning" in deep learning. Like with loss functions, different optimization functions are used for different types of networks, but they all contain the same two techniques at their core, backpropagation and gradient decent. Backpropagation is used to compute the gradient of the loss functions given the current network parameters, as seen in Eq. 5-5 [121,122]. Backpropagation uses the chain rule to start at the calculated loss score and compute the gradients of each previous layer due to its parameters. Once the gradient of the loss function is found, a gradient descent algorithm uses it to update the network parameters in the opposite direction of the gradient, as seen in Eq. 5-6 [121,122]. The learning rate in Eq. 5-6 is set by the user during training to optimize the results of the network. However, a learning rate too small might cause the network to get stuck on a local minimum, and a learning rate too large can cause the network to miss on important representations and features in the data [122]. The difference in optimizers for networks comes from the specific type of gradient descent function employed. The next section discusses which optimization function is used for the characterization algorithm.

$$MSE = \sum_{x \in X} (\mathbf{y} - \mathbf{y}')^2 \tag{5-4}$$

Where,

MSE = The mean squared error
$\mathbf{y}$ = The true values for inputs **x**
$\mathbf{y}'$ = The predicted values for inputs **x**

$$\nabla_p f(\mathbf{x}; p) \tag{5-5}$$

Where,

$f(\mathbf{x}; p)$ = The predicted values, **y'**, written as a function of inputs **x**
p = The network parameters, weights **w** and biases **b**

$$p_{i+1} = p_i - \eta \nabla_p f(\mathbf{x}; p_i) \tag{5-6}$$

Where,

p_i = The network parameters from the previous iteration
p_{i+1} = The updated network parameters
η = A constant referred to as the learning rate

To summarize, a neural network as seen in Figure 25 can be described with the following steps:

1. Define the training data **x** and its corresponding outputs **y**
2. Initialize random network parameters
3. Pass the input data **x** through the network to obtain the predicted outputs **y'**
4. Calculate the loss score by comparing **y** and **y'** with the loss function
5. Compute the gradient of the loss through backpropagation
6. Update the network parameters through gradient descent
7. Repeat steps 3-6 until an adequate loss score is reached

5.1.2. Convolutional Neural Networks

As was discussed in the previous subsection, each layer in a neural network is made up of several neurons. However, the neurons within a layer can be "stacked" in different ways to create different kinds of layers within a neural network that process data differently. The simplest type of layer is referred to as a Dense or fully connected layer. These types of layers are made up of neurons stacked in a single dimension and can, therefore, only process one-dimensional data. Figure 26 illustrates how a network of three Dense layers would look like. As can be seen in the figure, the first two layers have 8 neurons while the last layer only has one. The number of neurons in the last layer correspond to the number of outputs the network produces and is dependent on the objective of network. For example, a network trained to decide if a movie review is positive or negative should only give a single output, positive or negative (numerically represented as 0 or 1). As for the number of neurons in the previous layers, they correspond to the number of features learned from the data and are mainly found through trial and

error in most cases. However, there does exist an optimal number of neurons in Dense layers for each network. Too few neurons mean the network cannot learn enough features to correctly make predictions, and too many neurons mean that the network is learning features that do not contribute to making accurate predictions. It is also important to note that each Dense layer can have a separate activation function. As was mentioned in the last subsection, the most common activation function is ReLU(**y**). In most examples of neural networks nowadays, every layer, Dense or otherwise, does use ReLU(**y**) except for the last layer. The activation function for the last layer of a neural network depends solely on the objective of the network. For example, networks trained to output a regression may still use ReLU(**y**), but networks trained for classification usually use a sigmoid activation function, as seen in Eq. 5-7 and Figure 27. The sigmoid activation function outputs a probability between 0 and 1, which is ideal for binary classification problems, like the movie review example.

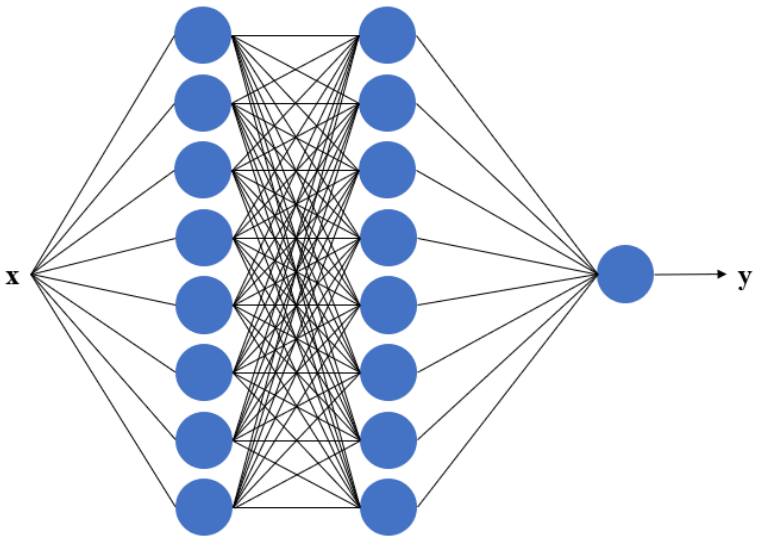


Figure 26. A visual representation of a three-layer Dense network.

$$sigmoid(\mathbf{y}) = \frac{1}{1+e^{-\mathbf{y}}} \tag{5-7}$$

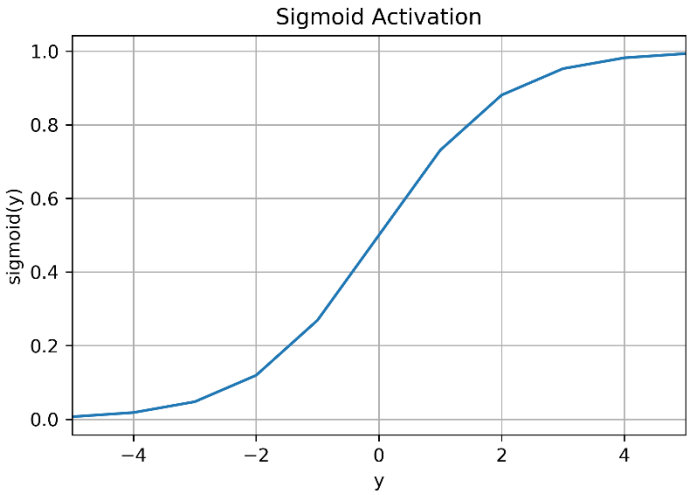


Figure 27. A plot of the sigmoid activation function given y**.**

Although networks only built from Dense layers can prove powerful for simple problems, neural networks developed for image classification, like the cat vs dog example or facial recognition, use what are referred to as convolutional neural networks (CNN). While Dense layers trained on images learn global patterns, CNNs learn local patterns through what are referred to as filters [121]. The advantage in using filters for learning local patterns is that the learned features are location independent. Once a pattern is learned from an object in an image, it can be recognized in another image in a completely different location. On the other hand, Dense layer learned features are restricted to the locations where they were first learned. This advantage allows CNNs to continually

outperform Dense layers in image identification tasks. For example, a CNN can be trained to recognize a car in an image by learning the patterns from headlights, tires, and side-mirrors. While Dense layers take in 1-dimensional data, CNNs operate on 3-dimensional data. The three dimensions include two spatial axes, like the height and width of an image, and depth axis [121]. When using images to train a CNN, the depth axis could be 3, for the red, blue, and green values of a colored image. Typical CNNs are made up of two types of layers, convolutional and max pooling layers.

In convolutional layers, filters pass through the input map of the data (the three dimensions previously mentioned) to extract features. Filters are typically of size 3 x 3 or 5 x 5, and they stop at every possible spot in an input map [121]. Figure 28 shows the patches of 3 x 3 data points in a 5 x 5 input map where the filter passes through [121]. The filter, or the convolutional kernel, performs a matrix multiplication and summation with the patch in the input data where it passes through to reduce the size of the input data; an example is also included in Figure 28. The result of the dot product becomes a data point on the outputted data map. Once the filter passes through all the patches, the outputted data map has smaller spatial dimensions. In Figure 28, the 5 x 5 input map transforms to a new map of size 3 x 3, spatially. With any convolutional layer, the spatial dimensions will always reduce by two. Since convolutional layers usually have multiple filters, the depth of the outputted data is transformed to the number of filters in the convolutional layer. Figure 29 shows a visual representation of how the dimensions change when input data of size 5 x 5 x 2 is transformed by a convolutional layer with three filters. Each filter

in Figure 29 operates like the example in Figure 28 to reduce a 5 x 5 map to size 3 x 3. The three resultant 3 x 3 maps are then stacked to form the new depth of the data.

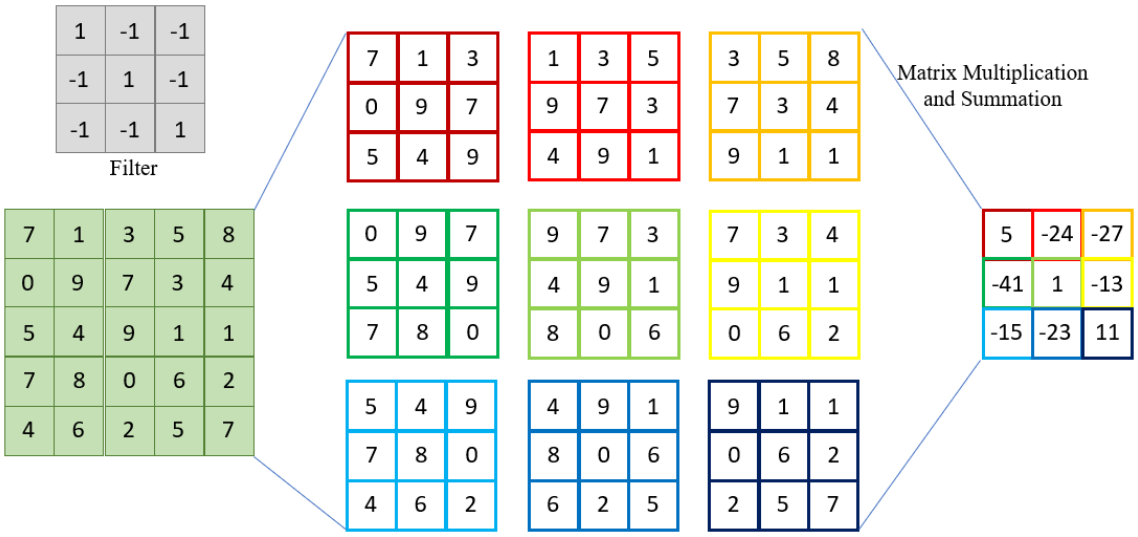


Figure 28. A visual representation of a 3 x 3 convolutional filter going through a 5 x 5 input.

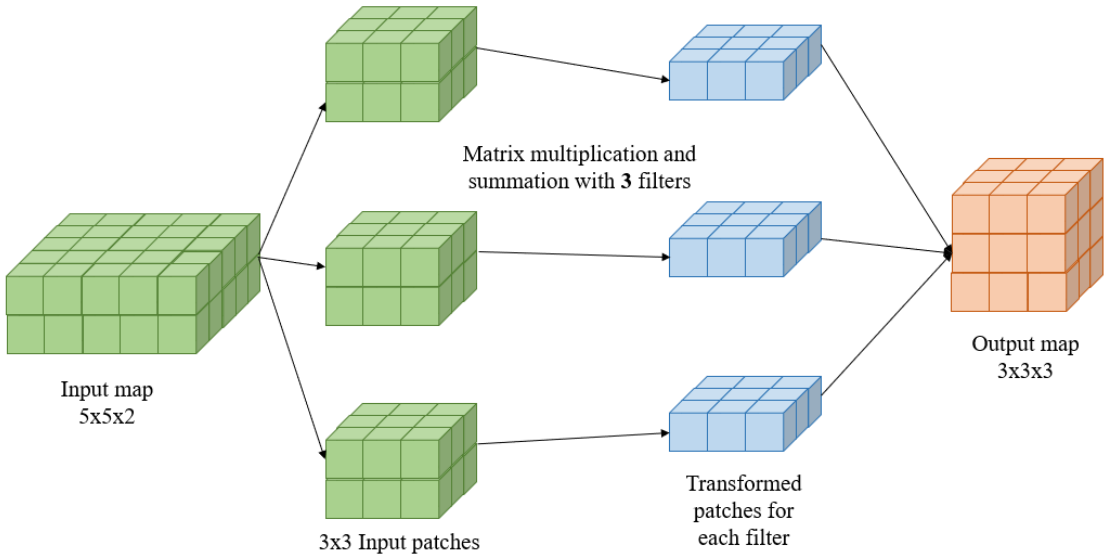


Figure 29. A visual representation of the dimension changes in a convolutional layer.

Max pooling layers are typically used in CNNs after convolutional layers to downsample the feature map. Downsampling is desired in CNNs to have less coefficients to process in feature maps and to have filters in later convolutional layers look at larger windows proportionally in relation to the feature map [121]. For example, if an input feature map of size 10x10 is put through two convolutional layers, its size will reduce to 8x8 and then 6x6. Proportionally, a 3x3 filter in the second convolutional layer will see more of the total feature map it receives than in the first layer. This increasing proportional filter window size is desired to create feature hierarchies in the network that can allow it to learn more abstract features. Functionally, max pooling layers operate in the same way as convolutional layers but utilize a "max" tensor operation instead of the convolutional kernel matrix multiplication and summation. Figure 30 conceptually illustrates the process in a max pooling layer. Only the largest number from each patch is chosen for the outputted data. While convolutional layers typically use filters of size 3x3 or 5x5, max pooling layers use filters of size 2 x 2. Figure 30 shows a 2 x 2 filter going through a 4 x 4 map that was outputted from a convolutional layer. It is important to note that the movement of max pooling filter is different than the convolutional layer. The filter in Figure 28 passed through every possible combination of 3 x 3, while the max pooling filter passes through independent sets of 2 x 2 pixels. This is referred to as stride. Convolutional layers typically use a stride of "1", which means the filter moves by only a single column or row. A max pooling layer stride of "2" means the filter moves by two columns or rows every time. As can also be seen in Figure 30, the dimensions of the map reduced from 4 x 4 to 2 x 2. When a max pooling layer is used in a CNN, it reduces the spatial dimensions

in the input feature map by half. If the spatial dimensions of input data are odd numbers, the resultant halved dimensions are rounded up to the nearest whole number. For example, data with size 7 x 7 will be outputted from a max pooling layer with size 4 x 4. Also, max pooling layers do not affect the depth of data, unlike convolutional layers.

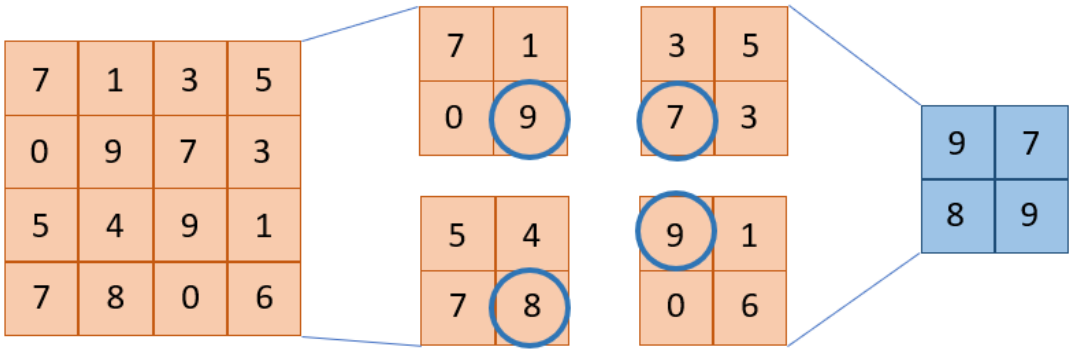


Figure 30. A visual representation of the Max Pooling layer.

Although convolutional and max pooling layers are the keystone of CNNs, in practice, the last layers of a CNN are Dense layers. After features have been extracted from the input data by convolutional and max pooling layers, the three-dimensional data representations are flattened and passed through Dense layers in order to accomplish the final task of the neural network. While the convolutional and max pooling layers essentially functioned as a dimensionality reduction technique, the final Dense layers complete the final classification or regression through the processes previously discussed. The next section describes the final neural network architecture for the characterization methodology.

5.2. Creating the Datasets

When creating a dataset for machine learning problems, it is essential to understand how the different ML algorithms work. Since the last subsection introduced the functionality of deep learning and neural networks, the discussion is now turned to the creation of the surrogate dataset. The first step in developing the surrogate dataset was determining the goals of the characterization methodology. As previously mentioned, the characterization methodology developed in this thesis falls under image classification given the objectives of the CubeSat surveillance system. Therefore, each data point in the surrogate dataset must represent an image containing phenomena of interest. With these "satellite images", the characterization methodology has a few options for its output. The characterization methodology could just identify if a scene captured by sensors is relevant or not, it could identify the individual phenomena in a scene, it could count the number of phenomena, or it could identify the location of targets within a scene. Each characterization objective requires a different neural network structure and different methods of creating the surrogate data set. Also, the number of neural networks used within the characterization methodology is another consideration. There could be a neural network trained for each phenomenon, or a network trained to identify all phenomena at once. It was decided that the characterization algorithm shall be created to identify if each phenomenon is present in a scene using the same neural network. This type of deep learning problem is referred to as multi-label classification [123]. In multi-label classification, the label for each data point or "image" in the surrogate dataset is a tensor with the same length as the total number of possible phenomena in a scene. Each number

in the tensor indicates the presence or absence of the phenomenon it corresponds to with a “1” or “0”, respectively. Outputting either a “1” or “0” from the neural network requires the sigmoid activation function previously discussed. Once the neural network is trained with the surrogate dataset, it will output the presence or absence of all phenomena of interest given a new “image”.

As was mentioned earlier, the creation of the surrogate dataset makes a few assumptions. First, each data point is a simple representation of a sensor image, not actual sensor data. The surrogate dataset is created in this fashion because of a lack of availability of sensor data from a CubeSat platform and to train the characterization methodology on data simple and general enough to allow for future adaptability to more specific data. Second, the dataset assumes that any data processing done between signal collection by the sensor and the production of a final image is already done. In reality, many data analysis techniques are employed for interpreting the signals received by sensors into images, like data fusion and improving signal-to-noise ratios. Even though the surrogate data set is a simple representation of satellite images, the sensor recommendations from the previous section were applied to the creation of the dataset. As can be seen in the last section, a total of four different sensor types were selected for observing phenomena of interest: panchromatic/multispectral sensors in the visible and near-infrared (VIS-NIR) range for dimensional parameters, multispectral sensors in the infrared (IR) range for temperatures, hyperspectral sensors in the IR range for gas emissions, and multispectral sensors in the ultra-violet (UV) range for aerosol measurements. To simulate the real-

world difference between all four sensors and their data, the surrogate dataset includes four separate datasets representing each sensor type.

5.2.1. Methodology

As previously mentioned, machine learning and deep learning model performance is highly contingent on the quality and quantity of data. Large quantities of diverse data are needed to ensure the robustness of these models. For a surveillance system on a CubeSat platform, identifying phenomena in a variety of scenarios is extremely important for accurate characterization and can be accomplished by training the characterization methodology on an adequate dataset. Therefore, the two overarching themes of the surrogate dataset creation are randomness and reproducibility. High levels of randomness when creating multiple data points ensures that the deep learning models learn important features solely relevant to the phenomena instead of learning situational features. For example, if in every "image" of the dataset there is always an object representing an automobile in the lower left corner, the deep learning model will get good at recognizing automobiles in the same location. However, if an automobile were to show up in the upper right corner of an image, the deep learning model might not recognize it since it related the automobiles in the training set to the lower left corner location. Therefore, a randomness in the placement of automobiles in the data allows the model to learn features exclusively pertaining to the automobile that can be recognized in any location or orientation in an image. Also, high reproducibility exposes the deep learning model to enough cases for statistical accuracy.

The creation of the surrogate dataset began with choosing the size of each "image". In other image recognition tasks in deep learning, the size of the input data refers to the number of pixels in the image. Images are represented as two to three dimensional tensors where each number represents the value of an individual pixel. For example, a black and white image could be of size 480 x 480 pixels where each pixel is a greyscale value. After some trial and error, a somewhat arbitrary number of 100 x 100 pixels was chosen for the size of the data in the surrogate dataset. An "image" of this size allows for multiple phenomena of different sizes in the image without seeming overcrowded. Each pixel value within the image represents either a phenomenon or background with different sized groups of pixels representing different phenomena as well. The overall dataset creation algorithm can be seen in Figure 31.

An image is first created by generating a 100 x 100 tensor of background values. The value for each pixel is created using Eq. 5-8. The variable *base* in Eq. 5-8 is where the difference in datasets representing the sensors comes in. Having different base values for the pixels simulates the different spectra where the sensors operate to collect data. Table 19 shows the different base values for the different datasets, the dataset names, and the sensors they correspond to.

$$background_{i,j} = random\_integer(0,9) + base \tag{5-8}$$

Where,

$background_{i,j}$ = The value for pixel in the *i* and *j* position
$random\_integer(0,9)$ =A random integer between 0 and 9
$base$ = The base value of the dataset type

Table 19. The different datasets, the sensors and spectra they correspond to, and their base value.

Dataset Name	Sensor	Spectral Region	Base Value
Dimension	Panchromatic/Multispectral	VIS-NIR	100
Temperature	Multispectral	IR	200
Gas	Hyperspectral	IR	300
Aerosol	Multispectral	UV	400

Eq. 5-9 shows a shortened example of an "image" with only background values in the Dimension dataset. The subscript next to each number in the tensor represents its coordinates in the *i* and *j* directions.

$$\begin{bmatrix} 100_{1,1} & 102_{1,2} & 108_{1,3} & \cdots & 107_{1,100} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 109_{100,1} & 104_{100,3} & 104_{100,3} & \cdots & 102_{100,100} \end{bmatrix} \tag{5-9}$$

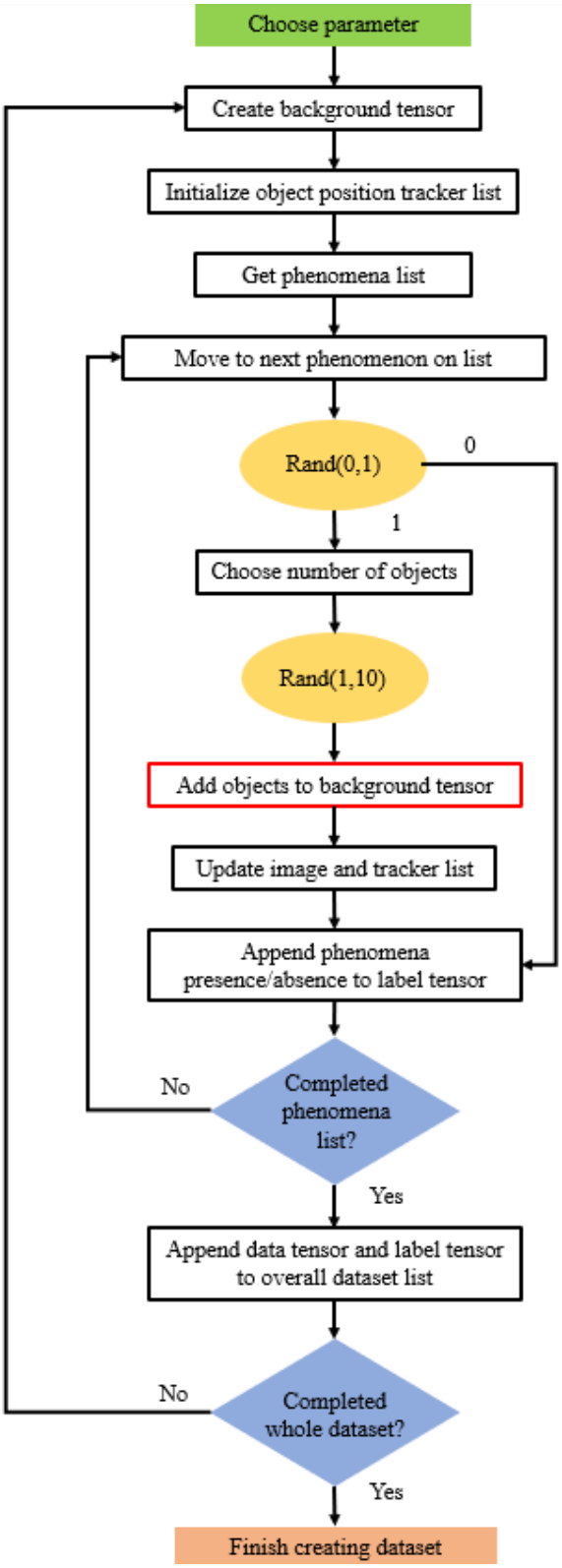


Figure 31. The algorithm for creating a dataset.

Once the background is created, a “tracker” list for object positions is initialized. The tracker prevents pixels of objects from overlapping when the objects are created. Whenever an object is added to the “image”, its *i* and *j* pixel coordinates are added to the tracker list. The tracker list has two dimensions, the i and j coordinates of every pixel of an object added and will have a length of the number of total pixels used for objects in the image.

Then, a list of phenomena observed by the detector corresponding to the dataset is retrieved. Tables 20-23 show the phenomena lists for each dataset, which were created from the sensor recommendations given for each parameter in Section 4. Each dataset is a type of parameter for each phenomenon, as was mentioned earlier. Pixel values and sizes are also included in the tables and are explained shortly. It is important to note that Tables 20-23 also include a “False Object”. The false objects create diversity in an image that is meant to challenge the characterization algorithm. They represent figures in the scene that are not of interest but are also not part of the background. Their inclusion prevents the characterization algorithm to learn phenomena features solely on pixel size, forcing it to also rely on phenomena values for characterization. As can be seen in the tables, the false objects can be of any pixel size but do have consistent pixel values that do not interfere with phenomena values. During the image creation algorithm, the false objects are treated the same as a phenomenon in the list, but their inclusion in an image is not specified with labeling. At the same time the phenomena list is retrieved, an empty list is created where the phenomena labels will be stored. For the list of phenomena, the algorithm goes through each phenomenon and decides whether to include it in the image by randomly generating

a “1” or a “0”. For the rest of this thesis, the function Rand(*a*,*b*) will refer to the process of randomly generating a number between the range *a* to *b*, inclusive. If Rand(0,1) gives “0”, the phenomenon is not included in the image, but if Rand(0,1) gives “1”, the phenomenon is included. If the phenomenon is not included, a “0” is added to the label for the image indicating it is not present. If the phenomenon is to be included, the algorithm then randomly picks a number between one and ten (Rand(1,10)) to represent how many instances of the phenomenon will be in the image and how many times the object addition sub-algorithm is initiated. The random inclusion of phenomena in random amounts in images ensures that the characterization neural network learns features independent to each phenomenon object instead of learning their presence by drawing correlations between different phenomena or the number of phenomena present. For example, without randomness, the neural network might learn to only recognize the presence of automobile object if there are also fire objects present in the scene. Through the use of the Rand operations, the neural network will identify phenomena in an image in any situation.

Table 20. The phenomena list for the Dimension dataset.

Phenomenon	Pixel Size	Pixel Value
Automobile	2, 4	40-44
Airplane	6	45-49
Facility	10	50-54
Construction/Mining Vehicle	4, 6	55-59
Mine Footprint	35	60-64
Fire	10	70-74
Blackout	10	75-79
False Object	2, 4, 6, 10, 24, 35	30-34

Table 21. The phenomena list for the Temperature dataset.

Phenomenon	Pixel Size	Pixel Value
Automobile	2, 4	40-44
Airplane	6	45-49
Construction/Mining Vehicle	4, 6	55-59
Construction/Mining Processes	10	80-84
Fire	10	70-74
Blackout	10	75-79
False Object	2, 4, 6, 10, 24, 35	30-34

Table 22. The phenomena list for the Gas dataset.

Phenomenon	Pixel Size	Pixel Value
Automobile	4, 6	40-44
Airplane	10	45-49
Construction/Mining Vehicle	6, 10	55-59
Construction/Mining Processes	10, 24	80-84
Fire	10, 24	70-74
False Object	2, 4, 6, 10, 24, 35	30-34

Table 23. The phenomena list for the Aerosol dataset.

Phenomenon	Pixel Size	Pixel Value
Construction/Mining Processes	10, 24	80-84
Fire	10, 24	70-74
False Object	2, 4, 6, 10, 24, 35	30-34

As can be seen in Figure 31, the box detailing when phenomena objects are added to the image is outlined in red, indicating a nested algorithm for adding the objects. The object addition sub-algorithm is the section of the overall algorithm where most of the randomness takes place and can be seen in Figures 32a-32c. As was previously mentioned, the accuracy of the deep learning model for characterization is increased when it is exposed to a large diversity of cases for the phenomena. The object addition sub-algorithm

ensures that a phenomena object is always placed randomly within the image with a random orientation. As can be seen in Tables 20-23, each phenomenon has a different pixel size and pixel value. The pixel sizes for each phenomenon were chosen in comparison to each other to represent the proportional difference in sizes between the phenomena. For example, a 2-pixel automobile represents a regular passenger vehicle, while a 4-pixel automobile represents a commercial truck. Since the surrogate dataset only aims at creating a simple representation of sensor images, the pixel sizes for phenomena objects are not necessarily proportional to spatial resolutions of sensors and their swath size. As can be seen in the tables, the automobile object is not the only one with different pixel sizes. For construction/mining vehicles, a 4-pixel object represents smaller equipment, like bulldozers and dump trucks, while a 6-pixel object represents larger equipment, like cranes. For construction/mining processes (like concrete mixing or blasting agents) and fires in the Gas and Aerosol datasets, the difference in sizes represents their gas emissions and aerosols moving through the atmosphere to encompass a larger area than the original object. As can also be seen in the tables, the pixel sizes also vary between datasets. For the Dimension and Temperature dataset, their pixel sizes are equal and represent the actual sizes of the objects. In the Gas and Aerosol datasets, however, the pixel sizes for the objects increase to the next tier up to represent gas and aerosol diffusion, as was just mentioned. As for the pixel values, different intervals of five numbers were assigned to each phenomenon. When adding the objects to an image, the *base* value for the dataset type is added to the pixel values as well. For example, an automobile in the

Dimensions dataset will have pixel values of 140-144, while in the Temperature dataset it will have values of 240-244.

As mentioned, the object addition sub-algorithm is completed for the number of times a phenomenon is added to the image from Rand(1,10). The algorithm starts by inputting the phenomena name and current tracker list. The phenomena name specifies the number of pixels to be used and the value attached to the pixels to represent the object, as seen in Tables 20-23. For the objects that have multiple pixel size, one size is randomly chosen. For the values for each pixel, a random number is chosen from the 5-number interval corresponding to the phenomenon and the dataset *base* value is added. Once the object size is decided, the coordinates for a corner of the object is randomly selected from the image index numbers, as seen in Eq. 5-10. The rest of the object pixels are extended from the randomly chosen corner.

$$\begin{aligned} i &= Rand(0,99) \\ j &= Rand(0,99) \end{aligned} \tag{5-10}$$

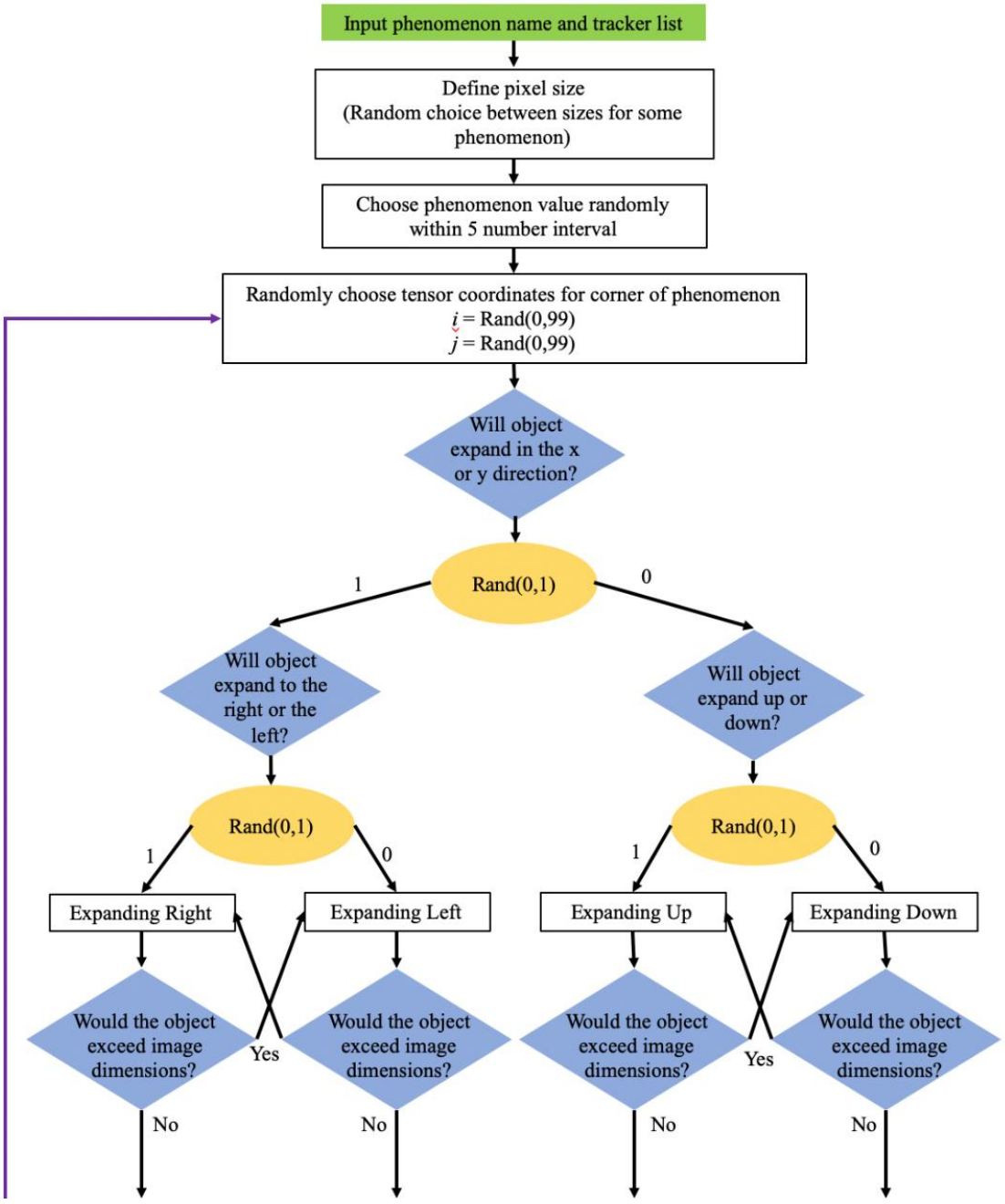


Figure 32. The first part of the object addition sub-algorithm.

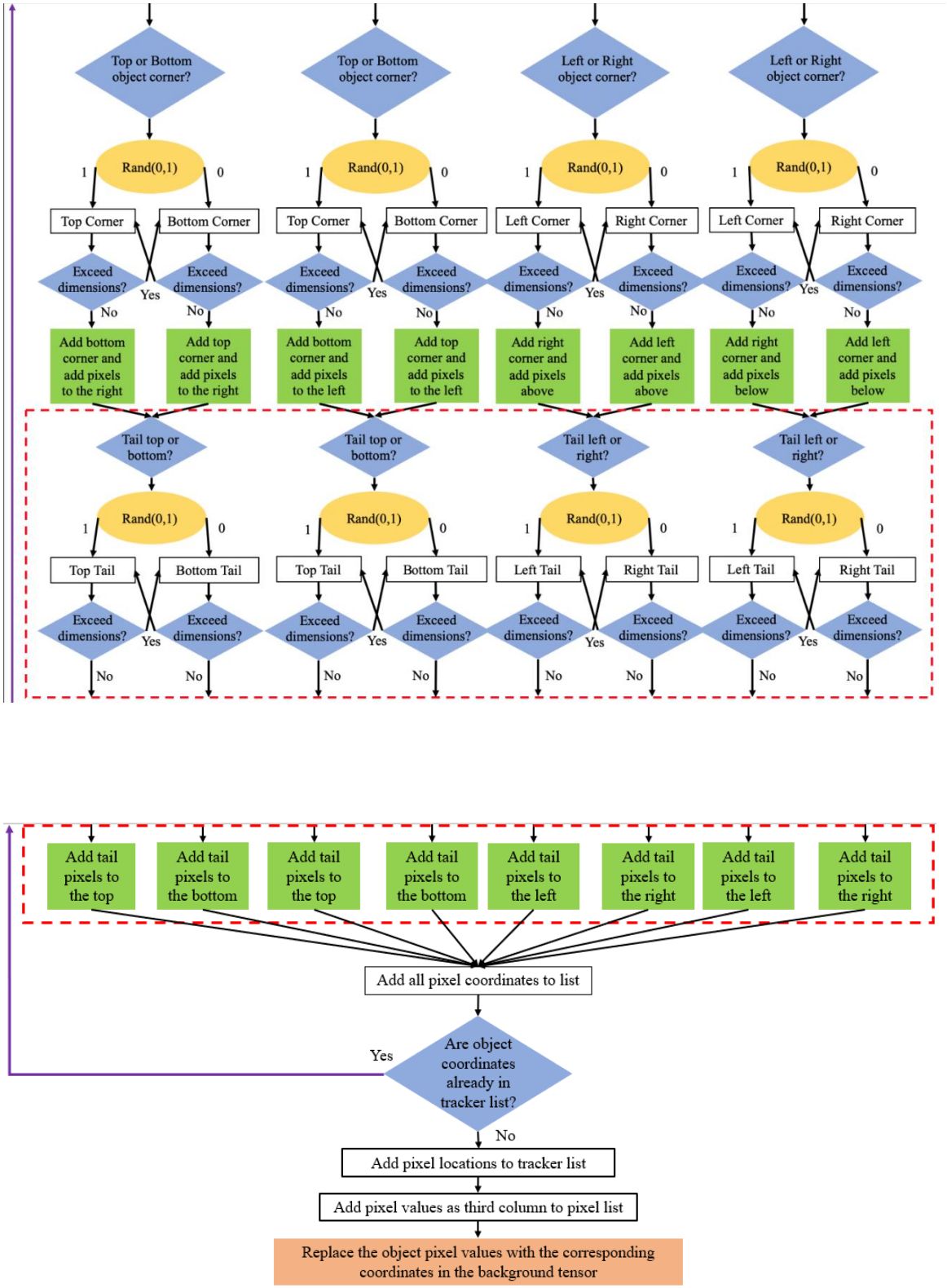


Figure 32. Continued.

Once the object corner is chosen, its orientation is chosen through Rand(0,1), a "1" indicating the object will extend in the x direction and a "0" in the y direction in the image. After the orientation is chosen, the direction is chosen again through Rand(0,1). In the x direction, a "1" indicates an extension to the right, while a "0" is an extension to the left. In the y direction, a "1" indicates an extension upwards, while a "0" indicates an extension downwards. While choosing the object direction in both orientations, a check is done to see if the object would exceed the image dimensions given its pixel size. The check is represented by Eq. 5-11. If the object exceeds the image dimensions, the opposite direction is chosen for the object. For example, if an object exceeds image dimensions by extending to the right, the direction is then chosen as left. Table 24 shows the different object lengths depending on pixel size, and Appendix B shows examples of the different objects. It is important to note that 24-pixel objects come in two versions. The different versions can be seen in Appendix B and the object addition sub-algorithm randomly choses which version to use when specifying an object's pixel size.

$$\begin{aligned} i-(object\ length-1)<0, i+(object\ length-1)>99 \\ j-(object\ length-1)<0, j+(object\ length-1)>99 \end{aligned} \quad (5\text{-}11)$$

Table 24. Object pixel length given pixel size.

Pixel Size	Object Length
2	2
4,6	3
10	4
24 (A)	8
24 (B)	6
35	7

After the final object direction is chosen for the object, a Rand(0,1) operation decides what corner within the object the previously chosen corner will constitute. In the x direction, a “1” indicates the chosen corner is the objects top corner, while a “0” is the bottom corner. In the y direction, a “1” indicates the chosen corner is the objects left corner, while a “0” is the right corner. Since each object is initially created with dimensions 2 x *object length* the chosen corner makes up one of the positions of the dimension of size two, and then the object is expanded by its length in the chosen direction. Figure 33 shows examples for the corner locations of a six-pixel object where the star marks the chosen corner. Once the object is extended in its chosen direction and has size 2 x *object length*, a “tail” is added to the object to give it a more interesting shape. The red box in Figures 32b and 32c indicates the section of the sub-algorithm where a “tail” is added to the object. For objects with no tail, the algorithm does not go through “tail” section. Only objects with pixel size 10 and 24 (version A) have tails. Also, objects of size 24 (version B) and 35 do not follow the 2 x *object length* dimensions. The 24 (version B) object is a rectangle of size 3 x *object length* that extends in the chosen direction after the corner location is specified, and the 35 pixel object has size 5 x *object length* that is also extended in the chosen direction after the corner location is specified. For adding the tail to the 10- and 24- (version A) pixel objects, the algorithm uses another Rand(0,1) operation. In the x direction, a “1” indicated the tail is added at the top of the object, while a “0” adds the tail to the bottom. In the y direction, a “1” adds the tail to the left of the object, while a “0” adds the tail to the right of the object. Like when choosing the object corner location, a check is done to ensure the object does not exceed image dimensions.

The checks are represented by Eq. 5-12. If adding the tail on one side of the object causes it to exceed image dimensions, the tail is therefore added to the other side. The $i \pm 1$ do the check for if the tail is added right next to the chosen corner and the $i \pm 3$ do the check for if the tail is added to the other side of the object width (which is two pixels). Once the side for the tail is chosen, a random position for it is chosen with Rand(0,2) for the 10-pixel object and Rand(0,3) and Rand(0,4) for the 24- (version A) pixel object. The tail for the 10-pixel object is of size 1 x 2, and the tail for the 24-pixel object is 2 x 4 but the position for each row of length 4 pixels is chosen separately, hence the two Rand operations. The Rand operations for the location of the tail are based on the location of the original corner. A "0" means the tail begins in the same row or column (depending on the direction of the object), while each subsequent number in the Rand range means the tail will begin the same number of rows or columns away from the original corner. For example, if a 10-pixel object with an original corner at (i, j) extending to the right, with the original corner as the top corner, and has a tail at the top of the object with Rand(0,2) = 1 for its position, the tail will have coordinates (i + 1, j + 1) and (i + 2, j + 1). Similarly, a 24-pixel object with the same orientation and corner conditions but with Rand(0,3) = 0 and Rand(0,4) = 2 for its tail position will have tail pixel coordinates of (i, j + 1), (i + 1, j + 1), (i + 2, j + 1), and (i + 3, j + 1) for the first row and (i + 2, j + 2), (i + 3, j + 2), (i + 4, j + 2), and (i + 5, j + 2) for the second row. Figure 34 shows visual representations of both examples. The orange stars in Figure 34 indicate the position of the original corner pixel, and the red stars indicate where the tail begins. The Rand operation range limit for the tail positions is decided by Eq. 5-13 and guarantees that a tail does not extend past the object's

overall dimensions. For the 24-pixel object, the difference in upper limits for each row to creates a more interesting figure since 24-pixel objects represent gas emissions and aerosols.

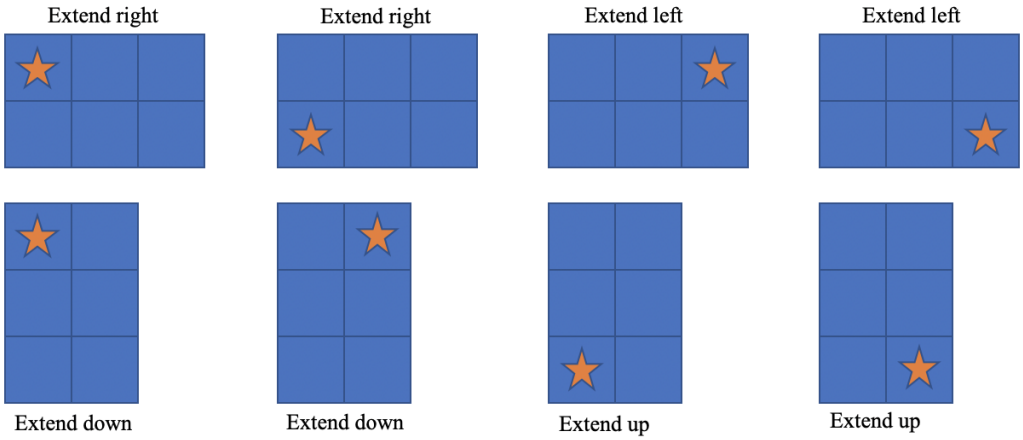


Figure 33. Corner and extension examples for a 6-pixel object.

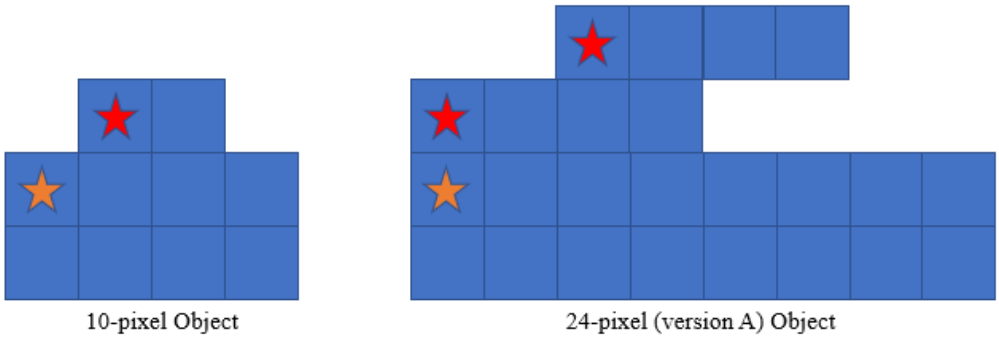


Figure 34. Examples of a 10- and 24- (version A) pixel object.

$$i-1<0\ or\ i-3<0, i+1>99\ or\ i+3>99$$
$$j-1<0\ or\ j-3<0, j+1>99\ or\ j+3>99 \quad (5\text{-}12)$$

$$tail\ position\ upper\ limit = \frac{object\ length}{2} \quad (5\text{-}13)$$

Once the tail coordinates are created, or once the tail section of the algorithm is skipped, all the pixel coordinated for the object are put in a list which takes shape 2 x *total number of pixels*. The object pixel list is then compared to the tracker list to see if any of the pixels overlap with previously created objects. If at least one pixel's location overlaps, the object addition sub-algorithm is reinitiated. The algorithm loops until an object is created with no overlapping locations. If no pixels overlap, the pixel locations of the object just created are added to the tracker list and then the pixel value representing the phenomenon chosen at the beginning of the sub-algorithm is added as a third column to every row of the pixel location list. The list is then outputted from the sub-algorithm and the pixel values of the image with the same coordinates as the object list are replaced by the object's pixel value. The object is therefore added to the image. Once an object is added to the image, the object addition sub-algorithm is repeated for however many objects of the same phenomenon are added (the Rand(1,10) operation from before). After all the phenomenon objects are added, a value of "1" is appended to the label list for the image, indicating the phenomenon's presence in the image. If the phenomenon was not added, a "0" is added to the label list. If the object which was added was a false object, no number is added to the label list. Their absence from the label list prevents the characterization algorithm to learn their features since they only represent obstacles for characterization. Figure 35 shows what a label tensor list might look like for an image in the Dimension dataset. It is important to note that the label list only indicated the presence of a phenomena, not the amount, as was decided by the objective of the characterization algorithm.

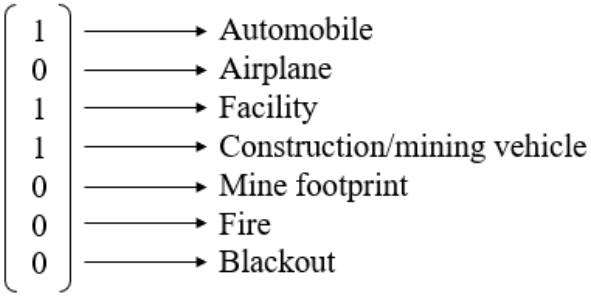


Figure 35. A label list for a data point in the Dimension dataset.

As can be seen in Figure 31, after a phenomenon is added (or not), the algorithm moves on to the next phenomenon in the dataset phenomena list to decide whether to include it or not, and the whole object addition sub-algorithm is repeated (or skipped). Once the algorithm passes through all the phenomena in the list, the final image and its label are added to the overall dataset list as one data point. The whole image creation process is repeated until the desired number of data points are created and added to the dataset. Examples of the dataset are seen in the next subsection.

5.2.2. Results and Discussion

Although examples of the different phenomena objects can be seen in Appendix B, the following subsections include examples of visual representations of images in each dataset. The following figures are only plots of the "image" tensor for better visualization. The characterization algorithm receives each data point in the form of the 100 x 100 tensor and a corresponding label tensor of shape 1 x *number of phenomena*. Since the characterization neural networks need to be trained on a high number of data points,

individual images are appended together and the input data to the network takes shape of *number of data points* x 100 x 100.

In the figures below, show visual representations of the created images. Each pixel in the grid represents a number in the 100 x 100 “image” tensor and its color represents the pixel value. The grid plots display the difference in features between phenomena that a CNN can easily learn. Each figure also includes the image’s label list indicating the presence or absence of each phenomenon. The tables below each figure show the number of each phenomenon present in the images. Even though only four images are included for each dataset as examples, the randomness in their creation is apparent. For example, Figure 43 includes all possible phenomena for the Temperature dataset, while Figure 50 does not include any phenomena at all. Also, some images include false objects while others do not. The possibility exists for one image to include ten objects for each phenomenon and the next image to have only background. Given the amount of Rand operations an integer values employed in the dataset creation algorithm, it is extremely difficult for two images to be the same. Training the characterization neural networks on datasets with so much diversity allows them to recognize each phenomenon in any situation.

5.2.2.1 Dimension Dataset

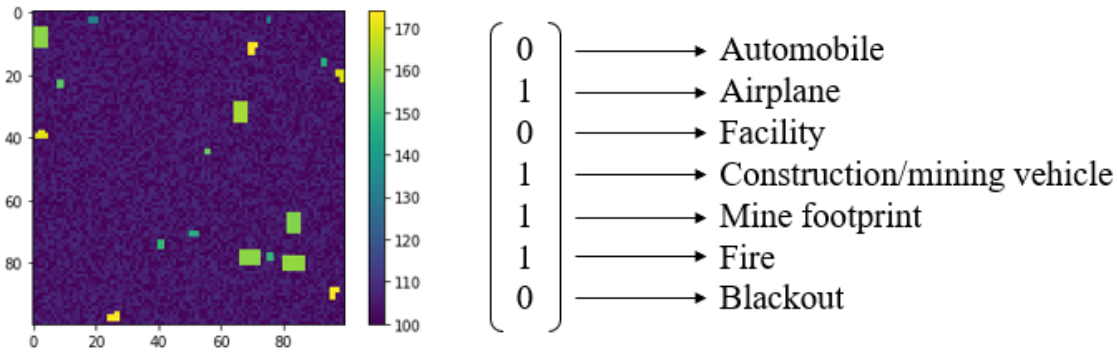


Figure 36. Example image one created in the Dimensions dataset.

Table 25. The number of each phenomenon present in Figure 36.

Phenomenon	Amount
Automobile	0
Airplane	4
Facility	0
Construction/Mining Vehicle	2
Mine Footprint	5
Fire	5
Blackout	0
False Object	2

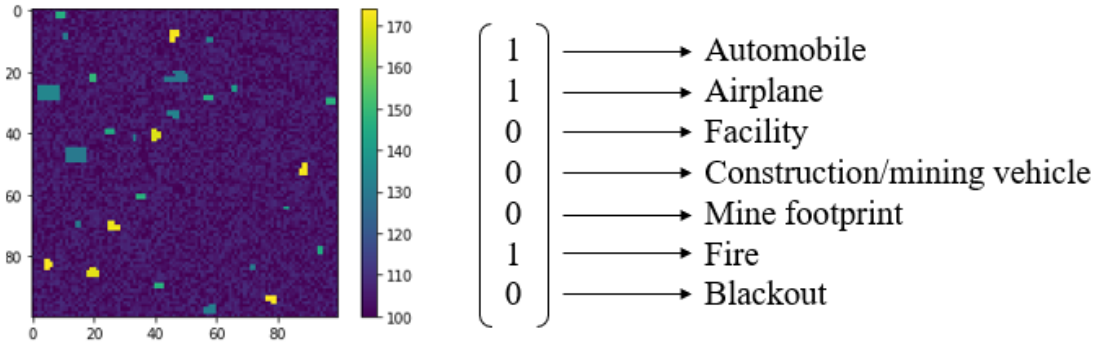


Figure 37. Example image two created in the Dimensions dataset.

Table 26. The number of each phenomenon present in Figure 37.

Phenomenon	Amount
Automobile	2
Airplane	8
Facility	0
Construction/Mining Vehicle	0
Mine Footprint	0
Fire	7
Blackout	0
False Object	10

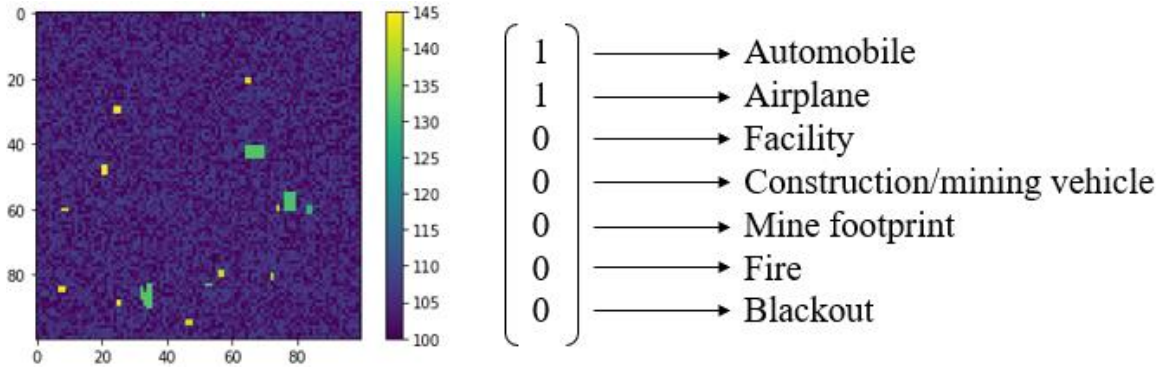


Figure 38. Example image three created in the Dimensions dataset.

Table 27. The number of each phenomenon present in Figure 38.

Phenomenon	Amount
Automobile	9
Airplane	1
Facility	0
Construction/Mining Vehicle	0
Mine Footprint	0
Fire	0
Blackout	0
False Object	6

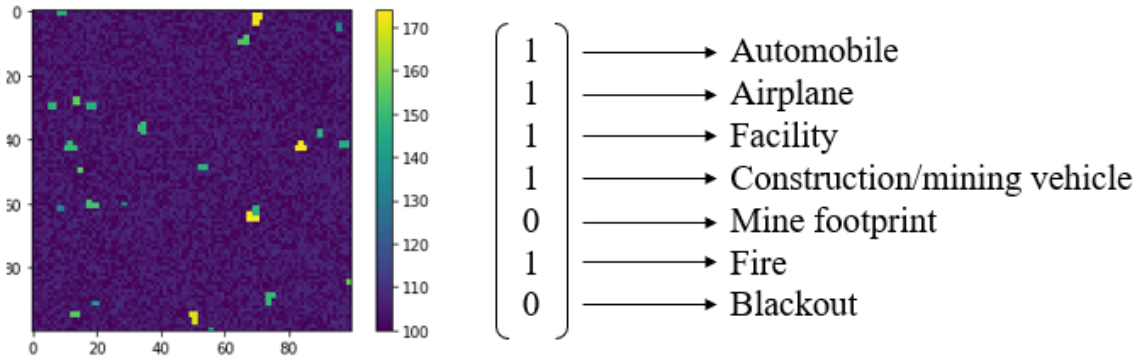


Figure 39. Example image four created in the Dimensions dataset.

Table 28. The number of each phenomenon present in Figure 39.

Phenomenon	Amount
Automobile	3
Airplane	7
Facility	5
Construction/Mining Vehicle	4
Mine Footprint	0
Fire	4
Blackout	0
False Object	2

The randomness in the dataset creation algorithm can be clearly seen in the difference between each "image", which were created consecutively. For example, Figure 38 only has two objects while Figure 39 is only missing two objects. It is important to also note the randomness present in the creation of the same type of phenomenon within the same picture. For example, each of the four fires in Figure 39 all have a different shape within the image. As mentioned, this randomness helps the characterization methodology algorithm learn features unique to phenomena that are not influenced by how they were placed in the image. The presence of "False Objects" can also be easily seen in the figures.

Even though Figure 38 only has automobiles and airplanes in the image, objects with a 24-pixel size which are not of interest can also be observed.

5.2.2.2 Temperature Dataset

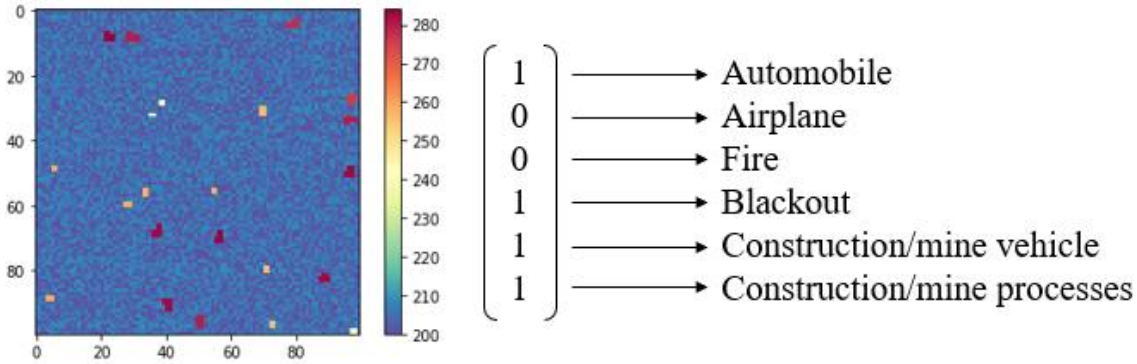


Figure 40. Example image one created in the Temperature dataset.

Table 29. The number of each phenomenon present in Figure 40.

Phenomenon	Amount
Automobile	3
Airplane	0
Fire	0
Blackout	2
Construction/Mining Vehicle	8
Construction/Mining Processes	9
False Object	0

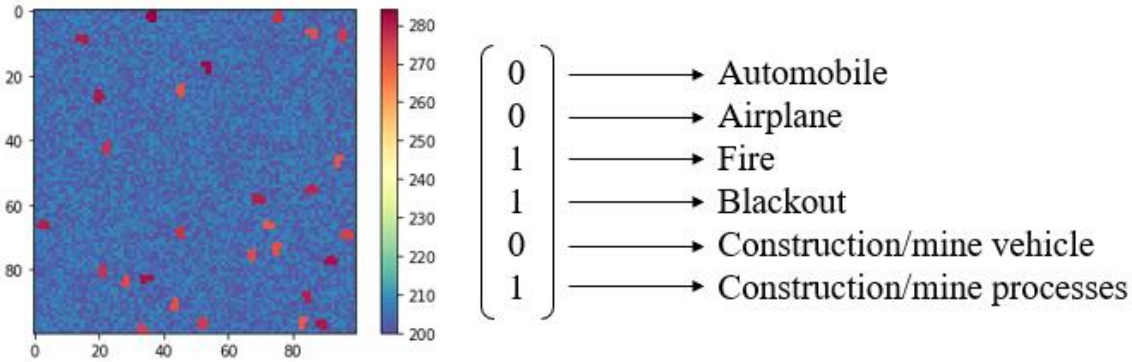


Figure 41. Example image two created in the Temperature dataset.

Table 30. The number of each phenomenon present in Figure 41.

Phenomenon	Amount
Automobile	0
Airplane	0
Fire	9
Blackout	9
Construction/Mining Vehicle	0
Construction/Mining Processes	10
False Object	0

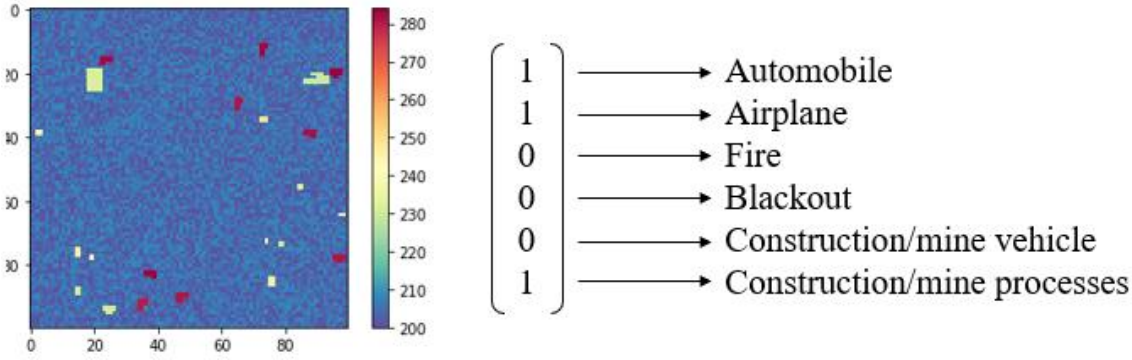


Figure 42. Example image three created in the Temperature dataset.

Table 31. The number of each phenomenon present in Figure 42.

Phenomenon	Amount
Automobile	4
Airplane	3
Fire	0
Blackout	0
Construction/Mining Vehicle	0
Construction/Mining Processes	9
False Object	6

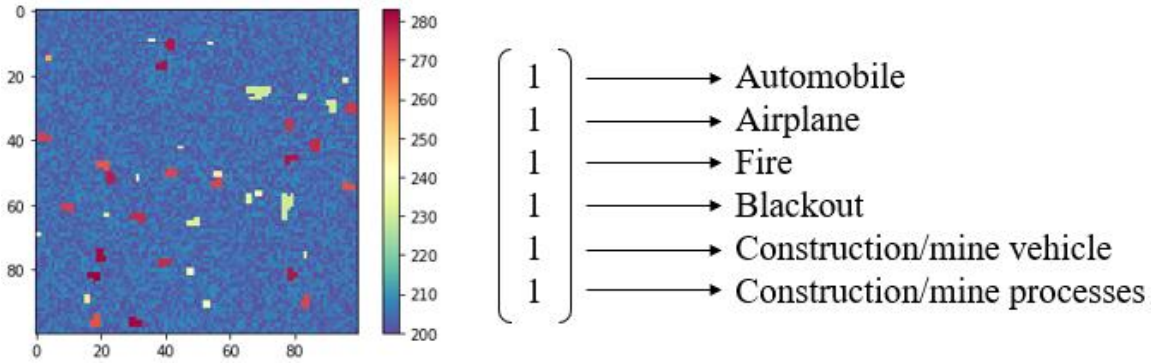


Figure 43. Example image four created in the Temperature dataset.

Table 32. The number of each phenomenon present in Figure 43.

Phenomenon	Amount
Automobile	8
Airplane	2
Fire	8
Blackout	6
Construction/Mining Vehicle	1
Construction/Mining Processes	7
False Object	9

Like with the Dimension dataset images, the Temperature dataset also produces highly randomized “images”. In contrast to other example “images”, Figure 43 includes every possible phenomenon. Even though it appears crowded, the dataset creation

algorithm prevented the objects in Figure 43 from overlapping. Unlike the other datasets, the maximum object size for the Temperature dataset is only 10 pixels. Figure 41 shows an "image" containing three out of six objects, all of size 10 pixels. Thanks to the difference in pixel value, the three phenomena are still distinguishable from each other. However, this fact of having half of the possible phenomena in the dataset all be the same size is the most likely reason for the characterization neural network's poorer performance on the Temperature dataset when compared to the other datasets, as is seen in Section 6.

5.2.2.3 Gas Dataset

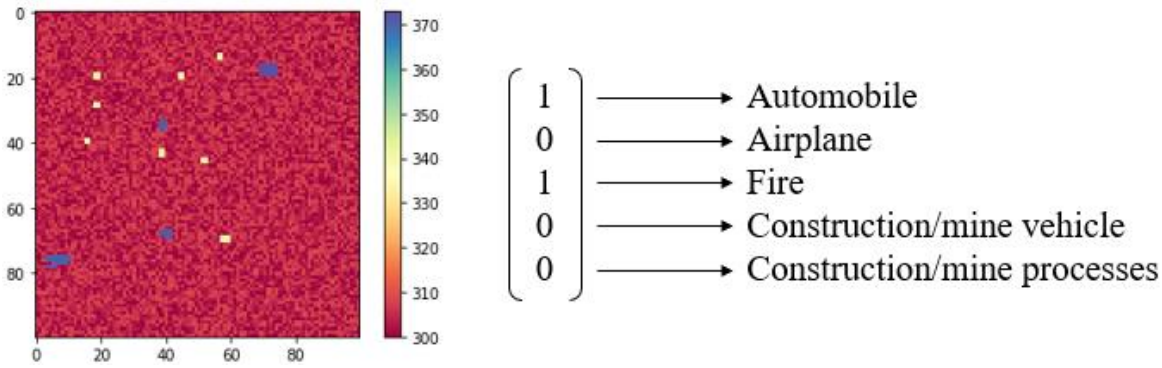


Figure 44. Example image one created in the Gas dataset.

Table 33. The number of each phenomenon present in Figure 44.

Phenomenon	Amount
Automobile	8
Airplane	0
Fire	4
Construction/Mining Vehicle	0
Construction/Mining Processes	0
False Object	0

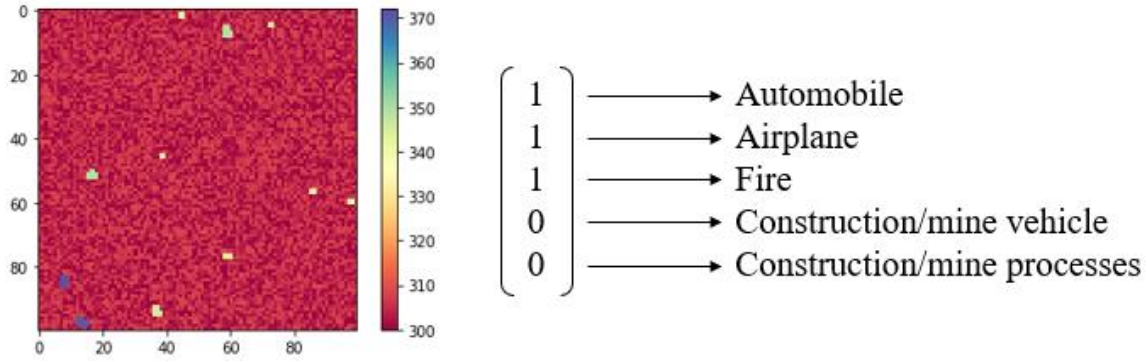


Figure 45. Example image two created in the Gas dataset.

Table 34. The number of each phenomenon present in Figure 45.

Phenomenon	Amount
Automobile	6
Airplane	3
Fire	2
Construction/Mining Vehicle	0
Construction/Mining Processes	0
False Object	0

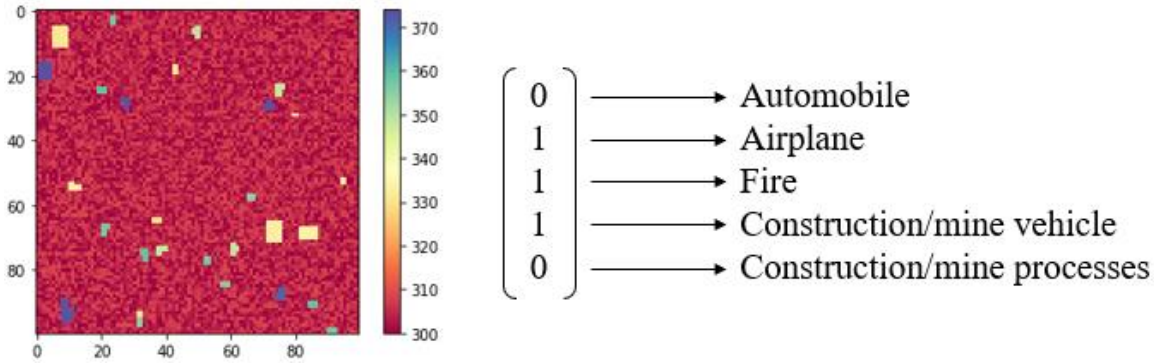


Figure 46. Example image three created in the Gas dataset.

Table 35. The number of each phenomenon present in Figure 46.

Phenomenon	Amount
Automobile	0
Airplane	4
Fire	5
Construction/Mining Vehicle	10
Construction/Mining Processes	0
False Object	9

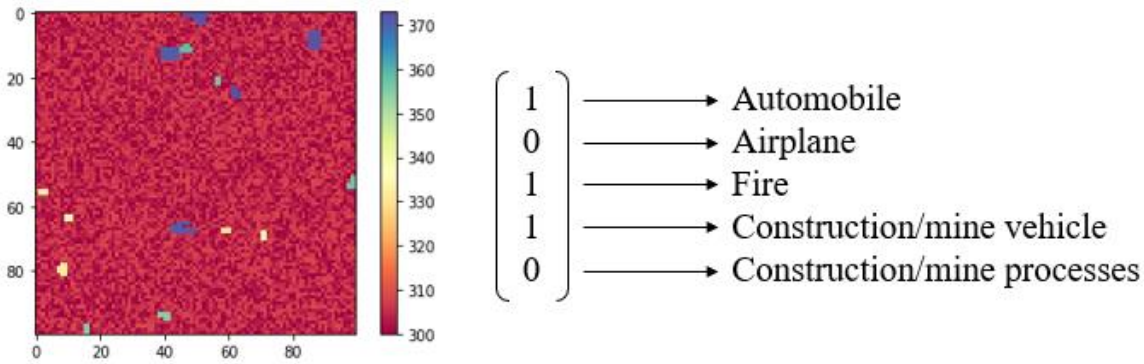


Figure 47. Example image four created in the Gas dataset.

Table 36. The number of each phenomenon present in Figure 47.

Phenomenon	Amount
Automobile	3
Airplane	0
Fire	5
Construction/Mining Vehicle	5
Construction/Mining Processes	0
False Object	2

Like with the other two datasets, the high levels of randomness within the dataset creation algorithm are present in the four Gas dataset example “images”. For example, Figures 44 and 45 have no “False Objects” within the image, while the rest of the other images seen so far all include them. The difference in individual phenomenon shape/size

can be easily observed with the fire objects in Figure 47. As was previously mentioned, the fire object for the Gas dataset is created in two possible sizes, 10 and 24 pixels, to emulate the propagation of gasses in the atmosphere. Fires with 10 and 24 pixels are present in the image, and the objects of size 24 also appear in the two different version shapes, A and B. As was mentioned, the Gas dataset has fewer observable phenomena than the previous two datasets, causing its images to appear a little less crowded.

5.2.2.4 Aerosol Dataset

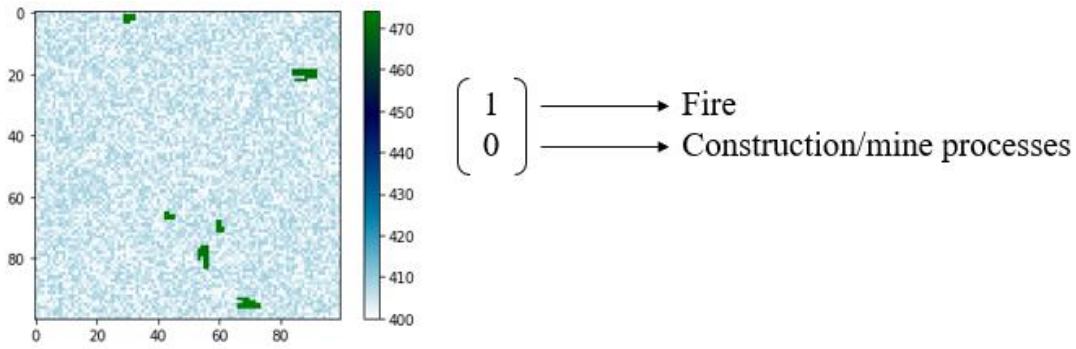


Figure 48. Example image one created in the Aerosol dataset.

Table 37. The number of each phenomenon present in Figure 48.

Phenomenon	Amount
Fire	6
Construction/Mining Processes	0
False Object	0

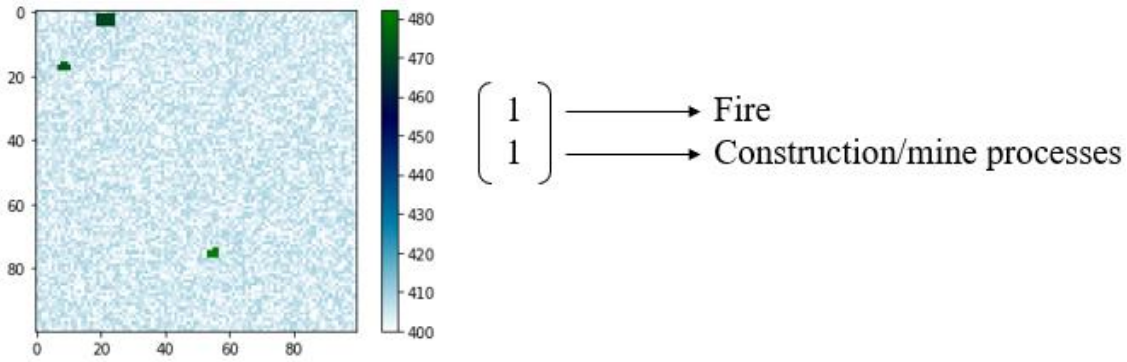


Figure 49. Example image two created in the Aerosol dataset.

Table 38. The number of each phenomenon present in Figure 49.

Phenomenon	Amount
Fire	2
Construction/Mining Processes	1
False Object	0

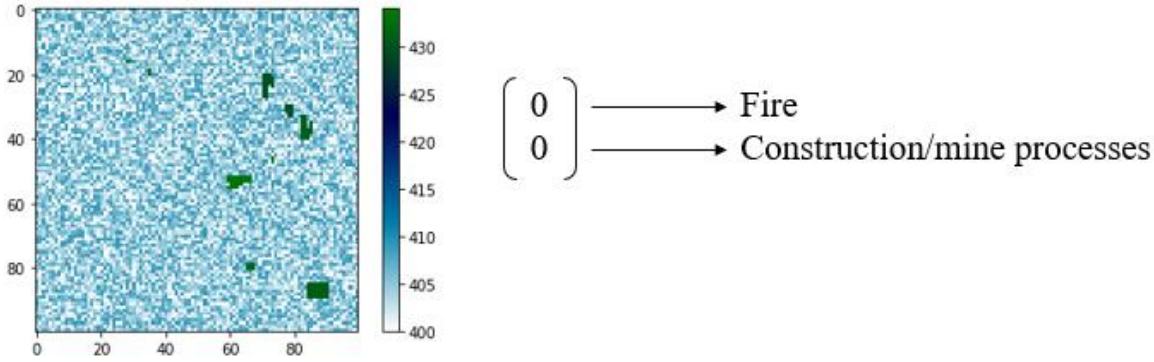


Figure 50. Example image three created in the Aerosol dataset.

Table 39. The number of each phenomenon present in Figure 50.

Phenomenon	Amount
Fire	0
Construction/Mining Processes	0
False Object	9

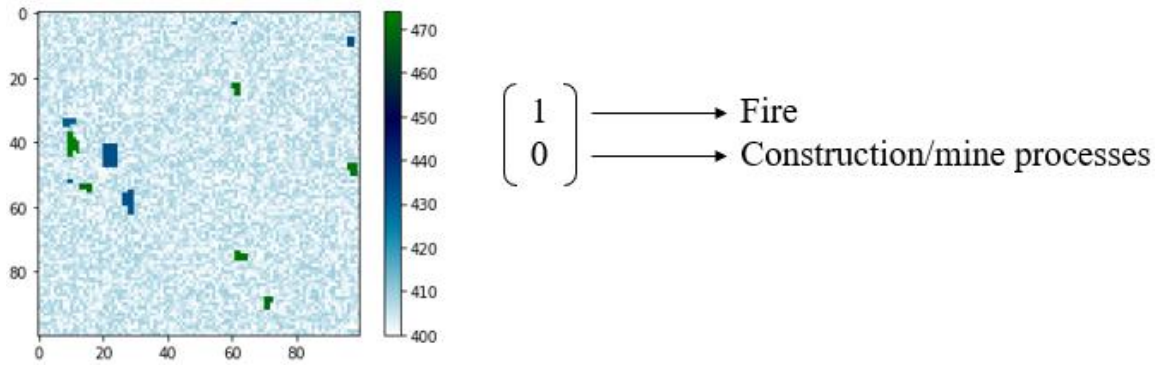


Figure 51. Example image four created in the Aerosol dataset.

Table 40. The number of each phenomenon present in Figure 51.

Phenomenon	Amount
Fire	6
Construction/Mining Processes	0
False Object	6

Finally, the Aerosol dataset example "images" also highlight the continued randomness in the dataset creation algorithm. For example, Figure 49 includes all possible phenomena while Figure 50 has none. The objects still visible in Figure 50, however, are only "False Objects". Figure 50 presents a clear example of how the characterization methodology is challenged by "False Objects". Just because there are objects in an "image", it does not mean they are of interest. Detecting the absence of phenomena is just as important as detecting their presence. It is also interesting to observe how the six uniquely looking objects in Figure 48 are all the same phenomena, as was also seen in the Gas dataset. Compared to the three other datasets, the Aerosol dataset has the least number of observable phenomena, causing the characterization methodology to produce its best performance on it, as is seen in Section 6.

5.3. Conclusion

This section served as an introduction to the concepts of machine learning and deep learning, covered the methodology used to create the representative surrogate dataset based on sensor types, and showed examples of the data. As was mentioned, the dataset creation algorithm is the same for each of the four datasets. The only differences are the *base* value added to each pixel and each dataset's corresponding phenomena list. Different phenomena lists create label lists with different lengths. As can be seen in Tables 20-23, the label lists are of length 7 for the Dimensions dataset, 6 for the Temperature dataset, 5 for the Gas dataset, and 2 for the Aerosol dataset. This difference in label list sizes and *base* values creates the need to have four different characterization models, one trained on each dataset. As is discussed in the next section, the characterization neural network has the same architecture for each dataset except for the last Dense layer which directly outputs the probability of a phenomenon's presence in an image in an output label list. Each number in the outputted list is a probability between 0 and 1 of a phenomenon's presence in the image.

6. CHARACTERIZATION METHODOLOGY

The previous section discussed the basics of deep learning techniques that are expanded in this section to create the characterization methodology for the surrogate dataset also discussed in the previous section. Although the same deep learning models created in this section cannot be directly applied to an actual CubeSat platform for characterization real sensor data, it provides a proof of concept and skeleton which can be easily adapted to different datasets as the surveillance platform continues development beyond this thesis. As was discussed during the creation of the surrogate dataset, each data point or "image" aims to provide a simplified representation of sensor data after it is already processed. The surrogate dataset and characterization methodology assume the data was already processed from sensor signals into recognizable images. The surrogate dataset includes four different datasets representing different sensor types and parameters. This section discusses the architecture developed for the convolutional neural networks (CNNs) trained on each of the four datasets. Although each CNN is trained on slightly different data, they have the same architecture of layers and functions. The training and optimization of the CNNs during their performance on new data are also discussed in this section. When the CubeSats in the constellation pass over a target, all sensors in the system are used to produce data or "images". In terms of the surrogate dataset, surveying a target produces an image in each of the four datasets. As each "image" is evaluated by its corresponding CNN model, each CNN outputs a probability for each phenomenon's presence or absence. The probabilities from each CNN from surveying a target are then combined to produce a final characterization solution. The last subsection provides an

illustrative application example of how each topic discussed thus far in the thesis is combined into a complete CubeSat surveillance system.

6.1. Model

As was mentioned in Section 5, the CNNs trained on the surrogate dataset form the main part of the characterization methodology. When the CubeSat surveillance system collects data from a scene on the planet's surface, the CNNs trained on data corresponding to each sensor directly make the characterization predictions for phenomena observed with each sensor (or dataset). However, the parallel CNNs must be brought together into an overall characterization methodology that can produce a final characterization solution of a target given the probabilities from each CNN. Figure 52 illustrates the structure of the overall characterization methodology and the data flow steps. Combining the phenomena probabilities into a final characterization solution, as can be seen in the last step of Figure 52, illustrates one of the advantages of having multiple satellites in the CubeSat surveillance system. A constellation of satellites allows for multiple sensors on board the overall system which give the characterization methodology access to multiple parameters for identifying phenomena. Using multiple parameters of a phenomenon to confirm its presence increases the methodology's characterization accuracy. Eq. 6-1 describes the last step of Figure 52 which combines each CNN's probability to produce a final solution. For each phenomenon, its probabilities of presence or absence in the scene from each CNN are averaged to create a final probability. As was seen in the previous section, not all datasets/CNNs include every possible phenomenon. For example, the aerosol dataset cannot detect the presence of a facility's physical building. As a result, only a

phenomenon's probabilities from the datasets in which it can be detected are averaged. To continue the example, a facility's final probability is not contributed to from the Aerosol CNN since it cannot be detected in that dataset in the first place.

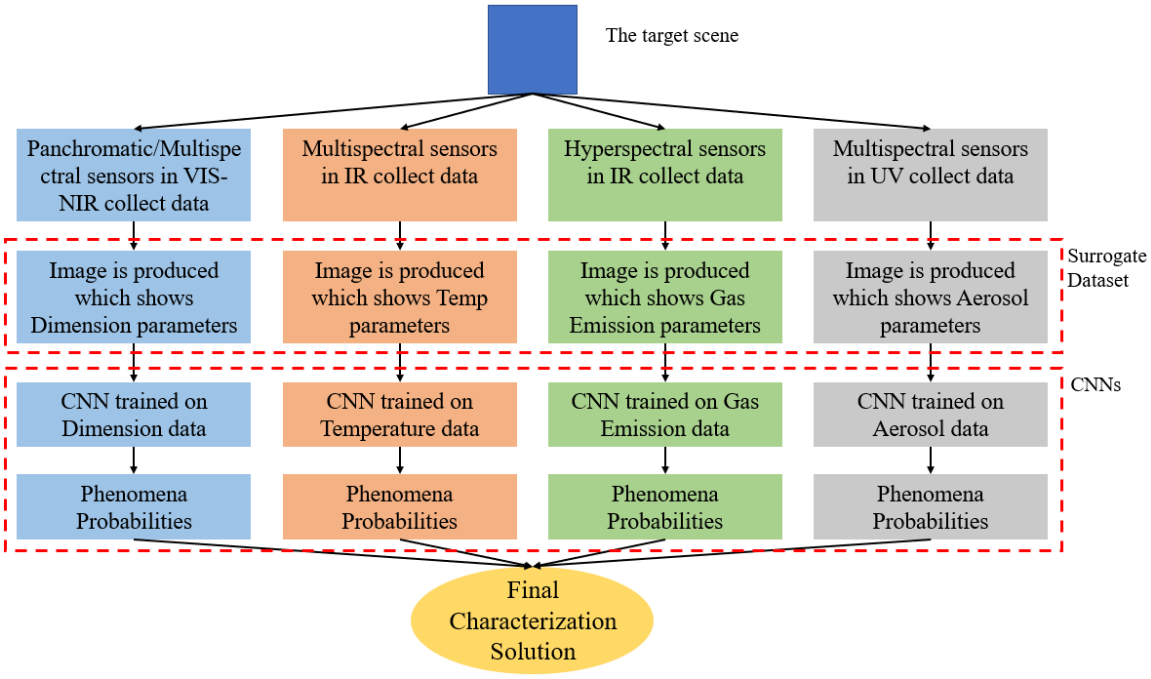


Figure 52. Structure of data flow in characterization methodology.

$$P(phenomenon_i) = [P(phenomenon_i|D) + P(phenomenon_i|T) + P(phenomenon_i|G) + P(phenomenon_i|A)]/N_p \quad (6\text{-}1)$$

Where,

$P(phenomenon_i)$ = Probability of presence in a scene for phenomenon i
D = CNN trained on Dimension dataset
T = CNN trained on Temperature dataset
G = CNN trained on Gas dataset
A = CNN trained on Aerosol dataset
N_p = Number of probabilities used for the calculation

Although the final step of the characterization methodology can be described by only one equation, describing each parameter in Eq. 6-1 is not as easy. Each probability is a result of a deep learning neural network trained on thousands of data points. The development of the four CNNs in the characterization methodology is discussed in the following subsections. As previously mentioned, each CNN has the same architecture, except for the last Dense layer of each. Since the CNNs are trained for a multi-label classification problem, the last Dense layer outputs the final probabilities for each phenomenon. Since the number of phenomena are different for each dataset, the last Dense layer for each CNN contains different numbers of neurons, or nodes. The number of nodes in the last layer directly correspond to the number of outputs from the network. It is important to mention that the development of the convolutional neural networks was accomplished using the Tensor Flow wrapper, Keras, within Python 3.7. The Keras module in python provides a simpler interface with the deep learning library. Also, the models were trained and evaluated on a GeForce GTX 950 with 1364 MB of memory.

6.1.1. CNN Architecture

Even though the CNN architecture went through an optimization process to achieve the best accuracy for the dataset, initial inspiration was drawn from networks in [121,124,125] which were used for image recognition tasks. The final CNN architecture can be seen in Figure 53. The architecture includes a total of 12 convolutional layers, with a max pooling layer after each group of three. All convolutional layers use filters of size 3x3, and the number of filters doubles with every subsequent group of three convolutional layers except for the last group which has the same number of filters as the group

preceding it. After passing through the convolutional and max pooling layers, the data is flattened in order to pass through the Dense layers at the end. The Flatten layer creates a 1-dimensional tensor with a length equal to the multiplication of the input data's three dimensions. Then, only one Dense layer, which includes 512 neurons, is used before the output layer. The number of neurons for the last Dense layer in Figure 53 is written as N because it varies depending on each dataset. The last layer has 7 neurons for the Dimensions dataset, 6 for the Temperature dataset, 5 for the Gas dataset, and 2 for the Aerosol dataset. Every convolutional layer in the network and the first Dense layer use the ReLU($\mathbf{y}$) activation function and the last Dense layer uses the sigmoid activation function. The nature of both activation functions is discussed in the previous section. When data is passed through the network, each data point or "image" tensor is introduced with dimensions (100, 100, 1) since the convolutional and max pooling layers only receive data in three dimensions. Using Python, the dataset images with dimensions (100, 100) are simply reshaped into size (100, 100, 1). Table 41 shows how the size of the data changes as it passes through the network. The table demonstrates how a convolutional layer transforms the third dimension of the input data into the same size as the number of filters. However, the convolutional layers in this network do not reduce the spatial dimensions of the input data by two, like how it was explained in the previous section. This is because all 12 convolutional layers implement padding in order to avoid losing information about the edges of the "image" as it passes through the network. Padding adds columns and rows on all four sides of the input feature map for the output dimensions of a convolutional layer to be the same as the input dimensions [121]. For filters of size 3 x 3, one row is

added to the top, one row to the bottom, one column to the left, and one column to the right, essentially making the input dimensions for the data (102, 102, 1). During training, the use of padding significantly increased the model's accuracy since object features on the edges of the image were not lost due to the convolutional layers. It can also be seen in Table 41 how the network transforms a tensor of size (100, 100, 1) into a single 1-dimensional tensor of size (N,) which included the probabilities for every phenomenon present in the input tensor. This illustrates how machine learning and deep learning techniques are used to reduce the size of data into meaningful representations.

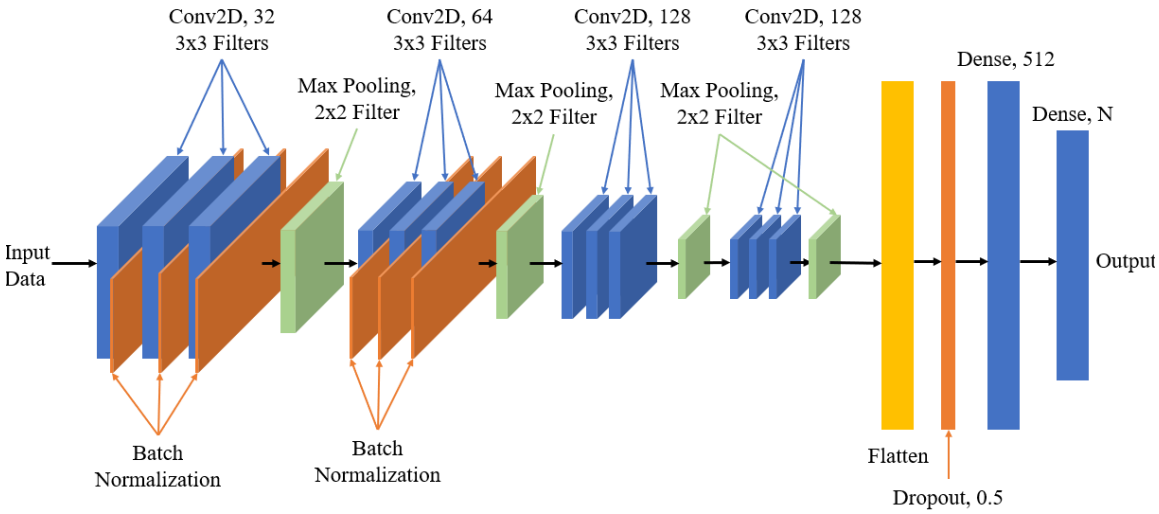


Figure 53. The final architecture for the convolutional neural network.

Table 41. The dimensions of the data after each layer in the CNN.

Layer	Output Shape
Conv2D 1	(100, 100, 32)
Batch Normalization 1	(100, 100, 32)
Conv2D 2	(100, 100, 32)
Batch Normalization 2	(100, 100, 32)
Conv2D 3	(100, 100, 32)
Batch Normalization 3	(100, 100, 32)
Max Pooling 1	(50, 50, 32)
Conv2D 4	(50, 50, 64)
Batch Normalization 4	(50, 50, 64)
Conv2D 5	(50, 50, 64)
Batch Normalization 5	(50, 50, 64)
Conv2D 6	(50, 50, 64)
Batch Normalization 6	(50, 50, 64)
Max Pooling 2	(25, 25, 64)
Conv2D 7	(25, 25, 128)
Conv2D 8	(25, 25, 128)
Conv2D 9	(25, 25, 128)
Max Pooling 3	(12, 12, 128)
Conv2D 10	(12, 12, 128)
Conv2D 11	(12, 12, 128)
Conv2D 12	(12, 12, 128)
Max Pooling 4	(6, 6, 128)
Flatten	(4608,)
Dropout	(4608,)
Dense 1	(512,)
Dense 2	(N,)

As can be seen in Figure 53 and Table 41, the network architecture also includes layers called “Batch Normalization” and “Dropout”. The inclusion of these layers in any network help reduce what is referred to as overfitting. The issue of overfitting is one that plagues most neural networks in deep learning, but it can be overcome through different techniques. When a model is training on a dataset, it uses a different subset of the data, commonly referred to as validation data, to test the performance of the model as it learns.

During training, the model processes the input training and validation data completely in multiple cycles called epochs. In one epoch, the model goes through every data point in segments called batches. As the model goes through each epoch, its performance on the training data improves as the neural network learns the representations of the training data. After each epoch, the model is evaluated on the validation data to test its performance on completely new data. Since the point of training neural networks is to produce accurate predictions on new data, the metrics produced by evaluating the validation data are what indicate the model's true performance, not the metrics produced from the training dataset. In an ideal model, the training and validation metrics, which are the network's loss and accuracy, should be close to the same value after each epoch. Overfitting occurs when the training metrics keep increasing with each subsequent epoch but the model's performance on validation data does not keep improving. This means that the model is learning irrelevant or misleading representations from the training data that does not translate to new data. When overfitting occurs, the best solution is to give the model more training samples. If using more samples is not possible or does not reduce overfitting, regularization techniques can be used [121]. As mentioned, the techniques used for this CNN model are batch normalization and dropout. Batch normalization helps reduce some overfitting in a network by increasing its stability. Increased stability prevents the performance on validation data to oscillate wildly, which decreases the reliability of a model. Batch normalization, essentially, adds noise to the previous layer's activation function, causing the model to only learn the features with higher importance [126]. It normalizes the output of the previous layer by subtracting the current batch's mean and

dividing by its standard deviation. The batch normalization process can be seen in Eq. 6-2 – 6-5 [126]. The parameters γ and β are learned in the training process through gradient descent, like the weights and biases in the other layers. During the training of the model, it was found that with each batch normalization layer added, the model's performance increased, and overfitting reduced. After the addition of the sixth batch normalization layer, improvements in performance and overfitting plateaued. The number on batch normalization layers was therefore left at only six since each additional layer causes fewer total features to be learned from the data.

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}_i \tag{6-2}$$

Where,

$\mu_B =$ The batch mean
$m =$ The batch size
$\boldsymbol{x}_i =$ The outputs of the previous layer to be normalized within the batch

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{x}_i - \mu_B)^2 \tag{6-3}$$

Where,

$\sigma_B^2 =$ The batch variance

$$\widehat{\boldsymbol{x}_\iota} = \frac{\boldsymbol{x}_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{6-4}$$

Where,

$\widehat{\boldsymbol{x}_\iota} =$ The normalized mean of the data in the batch
$\epsilon =$ Batch normalization constant added for stability

$$BN_{\gamma,\beta}(\boldsymbol{x_i}) = \boldsymbol{y_i} = \gamma\widehat{\boldsymbol{x}_\iota} + \beta \tag{6-5}$$

Where,

$BN_{\gamma,\beta}(\boldsymbol{x_i}) = \boldsymbol{y_i} =$ The normalized output of the previous layer
$\gamma =$ Learned "mean" parameter
$\beta =$ Learned "standard deviation" parameter

The dropout layer functions like batch normalization in the sense that it prevents the model from learning insignificant features from the data. When applied to a layer,

dropout randomly sets a certain amount of the output's values to zero during training [121]. The dropout rate can be set manually by a user which refers to the fraction of value in a layer's outputs set to zero during training. For example, a dropout rate of 0.5 sets half of a layer's outputs to zero. During testing, the dropout added to a layer does not "zero out" the outputs like in training. Instead, the output is scaled down by multiplying by the dropout rate to maintain the same number of values present in the output of the layer as during training. Randomly setting values to zero prevents the neural network from learning insignificant features in the data. The location for the dropout added to the model in this thesis, as seen in Figure 53, was inspired by the networks in [121,125].

Figure 54 below shows an example of a neural network overfitting during training, while Figure 55 shows an example of a model with no overfitting. These graphs were produced during the optimization process for the CNN, so their accuracies do not reflect the final model training curves. As can be seen when comparing the different sets of figures, the validation curves for accuracy and loss diverge from the training curves for the overfitted model. It is important to note that even though some spikes in the validation values in Figure 55 occurred, they do not indicate the presence of overfitting. The presence of spikes means that convergence during training was not entirely "smooth". Although a few large steps towards convergence (spikes) were taken during the gradient descent process, the curves in Figure 55 do display convergence. However, when choosing between two models with similar performance during an optimization process, the model displaying smoother validation curves, or better convergence, is preferable as its final test performance will display more consistency if it is trained multiple times.

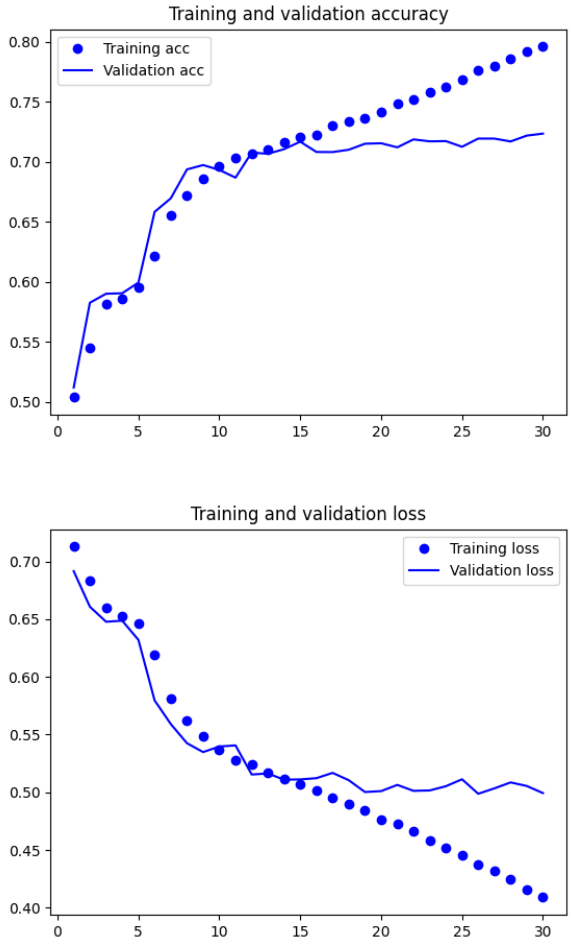


Figure 54. The training and validation accuracy and loss for an overfitted model.

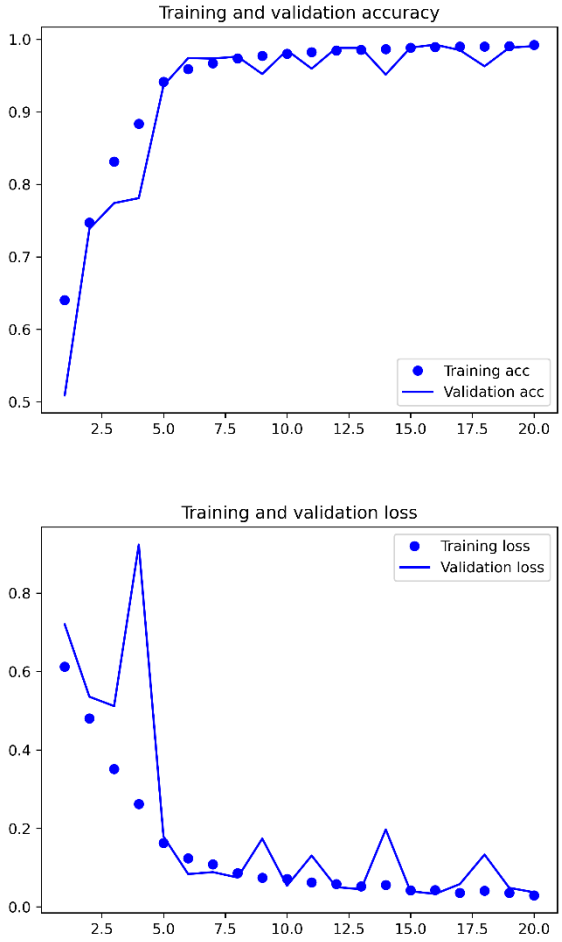


Figure 55. The training and validation accuracy and loss for non-overfitted model.

6.1.1.1. Optimization

While constructing the final CNN architecture, a few different parameters were changed to find an optimally performing network. Those parameters were: number of Conv2D layers, number of Dense layers, number of filters for the Conv2D layers, filter size, and number of neurons for the Dense layer(s). During the optimization of the architecture, only one parameter was changed at a time in order to observe its true effect on the network's performance. Also, the hyperparameters were kept the same while

training each model. Hyperparameter optimization is discussed in the Training subsection. For evaluating the performances of each network architecture, their accuracies on a set of test data are compared. The performance accuracy for a network is the percentage of data points which it predicted correctly. It is also important to mention that during the optimization of the CNN, only the Dimension dataset was used. Once a final architecture was found, only the number of nodes in the last Dense layer of the network were changed when training with the other three datasets. It is important to note that while training the different models in the optimization process, a Keras "callback" was used to only save the best performing model during training instead of the model produced at the end. For example, if the validation accuracy after 15 out of 20 training epochs is 96% and that score is not reached again in the remaining epochs, the model after the 15^{th} epoch is saved as the final model. Saving the best model while training prevents the final network performance from suffering if spiking or overfitting occur towards the later epochs. Using the best model during training instead of the final one illustrates the full potential of a certain architecture during optimization. Also, during the optimization processes, individual architectures were trained several times in order to understand the full scope of their performance. Due to the random nature of initializing weights during training, a model may not produce the same performance every time it is trained. It was observed that the final model accuracies for individual architectures after several training sessions had a standard deviation of 0.26% accuracy. The error bars included in all the architecture optimization graphs reflect this 0.26% training variation for architectures.

When optimizing for the number of Conv2D layers to include in the architecture, the idea of four-layer groups separated by max pooling layers persisted through each iteration. Four different architectures were trained and evaluated containing 4, 8, 12, and 16 convolutional layers. Figure 56 shows how accuracy is affected by the number of Conv2D layers. As can be seen, the test accuracy of the network increases with the number of layers, with the 12- and 16-layer architectures producing accuracies within error. The 4- and 8- layers models achieved lower performances since they were not able to learn enough features to successfully predict the data due to a lack of layers. Since the 12- and 16-layer architectures produced the same test accuracies and showed the same amount of spiking in the training and validation curves, the 12-layer model was chosen as the final architecture to save on computational resources during training. For the Dense layers in the network (not including the final layer), optimization began with a single layer and each subsequent iteration added an extra layer (while still before the final layer). It was seen that the inclusion of a second Dense layer did not significantly improve the model's accuracy and decreased the smoothness during training, so only one Dense layer was chosen for the final architecture.

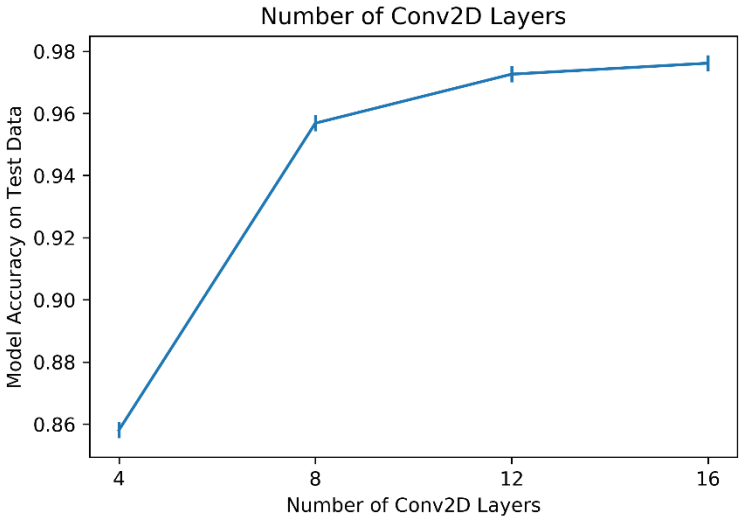


Figure 56. Model accuracy on test data depending on number of Conv2D layers.

With the number of layers chosen, optimization for the number of filters for the Conv2D networks and number of neurons in the Dense layer was done. For the Conv2D layers, the same trend of increasing the number of filters by a factor of two with each new convolutional 3-layer group (except for the last group of layers) was kept. The different filter numbers used for the first convolutional layer group were 16, 32, and 64. The second layer group of convolutional layers for each iteration then has 32, 64, and 128 filters, respectively, and then the third and fourth groups have 64, 128, and 256 filters, respectively. It is important to note that the number of nodes for the Dense layer were set to 512 for each of the three models. Figure 57 shows the final model accuracy for each of the three architectures. Model test accuracy increased as the number of filters in the Conv2D layers increased. The difference in test accuracy errors between the 32 and 64 initial layer filter architectures was only 0.2%.

Even though the architecture with 32 initial layer filter architecture produced slightly lower test accuracies, it was chosen for the final model since the 64 initial layer filter architecture produced displayed more overfitting in the training curves and was more computationally expensive. As mentioned, the presence of spiking or slight overfitting does not necessarily affect a model's final accuracy performance on a test dataset. However, it can affect a model's performance on predicting individual phenomena in "images", as is discussed in the Results and Discussion subsection. As a result, when choosing between two models with similar test accuracies (less than 0.5% difference), the model with less spiking and overfitting is preferred. For the number of nodes in the Dense layer, 128, 256, 512, and 1024 nodes were used. For all four architectures, the number of filters for the Conv2D layers were set to 32, 64, 128, and 128 for the four-layer groups. Figure 58 shows that the architecture's test accuracy peaked at 512 nodes, with its accuracy performing within error of the 256-node model. As the number of nodes increased, more representations from the data were learned, which increased test accuracy. However, the 1024-node model produced lower performance due to overfitting. Other than producing test accuracies within error, the 256- and 512-node architectures also displayed the same frequency of spiking in the training and validation curves. The 512-node architecture was chosen as the final model to allow the network to learn more representations from the data.

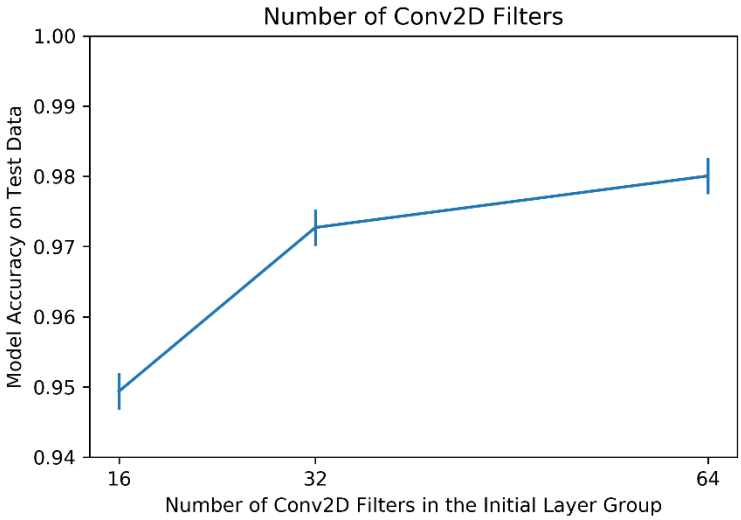


Figure 57. The final model accuracy when compared to number of Conv2D filters.

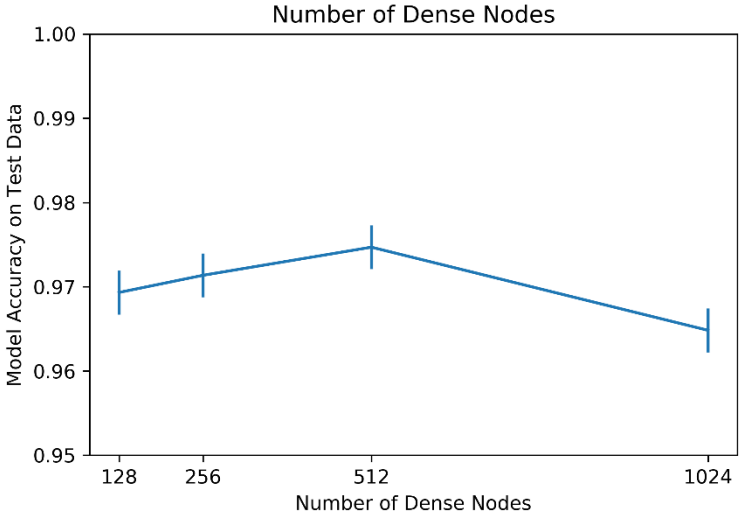


Figure 58. The final model accuracy when compared to number of Dense nodes.

The last architectural feature to be optimized was the filter size for the convolutional layers. The different sizes used were 3 x 3, 5 x 5, and 7 x 7. As can be seen in Figure 59, the architecture test accuracy decreased as the filter size increased. Even

though it is unclear in Figure 59, the error bars for the 3 x 3 filter model and the 5 x 5 filter model do not overlap. Also, overfitting was more prevalent as the filter sized increased. Therefore, the final filter size for the Conv2D layers was chosen as 3 x 3.

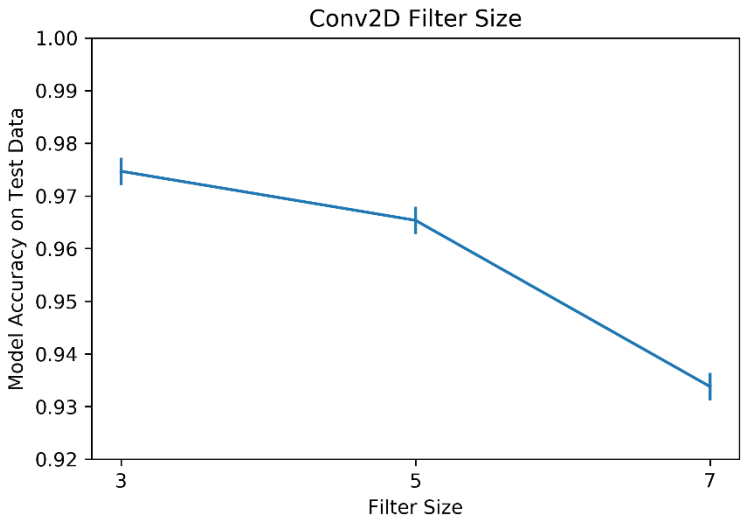


Figure 59. The final model accuracy when compared to changing Conv2D filter size.

6.1.1.2. Compiling the Model

Once a final architecture is chosen for a CNN, it must be compiled within Keras by specifying the optimizer for the network, the loss function, and any metrics for calculation other than the loss score. As was discussed in the previous section, the loss function measures the performances of the neural network, while the optimizer updates all the network parameters through gradient descent techniques. The loss function implemented into this CNN is referred to as binary cross-entropy. Typically, the loss function for a neural network is determined by the type of output, or measure, desired from

the model. Since this CNN aims to solve a multi-label classification problem which outputs the probability of a phenomenon, binary cross-entropy is a natural fit. This loss function, as described with Eq. 6-6, computes the loss for the predicted value out of two labels. It multiplies the log probability of the predicted correct label with the true correct label and adds it to the multiplication of the log probability of the opposite of the predicted label with the opposite of true label. Eq. 6-7 give an example of a calculated loss score for a predicted label of 0.9 when the true label is 1 for a single sample. As can be seen, the loss is calculated as 0.1054. If the predicted label would been a very accurate prediction of 1, the loss value would be 0. Once a loss score is calculated for a set of data, the gradient descent algorithm in the optimizer aims to reduce it in the next iteration by updating the network parameters.

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(y_i') + (1 - y_i)\log(1 - y_i') \tag{6-6}$$

Where,

$Loss =$ The loss value
$N =$ The total number of samples
$y_i =$ The true label value (between 0 and 1)
$y_i' =$ The predicted label value (between 0 and 1)

$$-(1 * \log(0.9) + (1 - 1) * \log(1 - .9)) = 0.1054 \tag{6-7}$$

The optimizer chosen for this CNN is called RMSprop, a popular choice in the world of deep learning. When compared to other optimizers, RMSprop scales the learning rate in gradient descent by a moving average of the squared gradient [128,129], as described in Eq. 6-8 and 6-9. It is worth noting that Eq. 6-8 is only a slight alteration from Eq. 5-6 in the previous section. The learning rate in Eq. 6-8 is a hyperparameter which was optimized during training of the CNN. Another popular optimizer, Adam, was also

explored, but RMSprop was chosen at the end of the day because it produced the same results at a much higher learning rate, meaning the network trained faster.

$$p_{i+1} = p_i - \frac{\eta}{\sqrt{E\left[\nabla_p f(\mathbf{x}; p_i)^2\right]}} \nabla_p f(\mathbf{x}; p_i) \tag{6-8}$$

Where,

$p_i =$ Current network parameters
$p_{i+1} =$ New updated network parameters
$\nabla_p f(\mathbf{x}; p_i) =$ The gradient of the network
$E\left[\nabla_p f(\mathbf{x}; p_i)^2\right] =$ The moving average of the squared gradient
$\eta =$ The learning rate

$$E\left[\nabla_p f(\mathbf{x}; p_i)^2\right] = \gamma E\left[\nabla_p f(\mathbf{x}; p_{i-1})^2\right] + (1 - \gamma) * \nabla_p f(\mathbf{x}; p_i)^2 \tag{6-9}$$

Where,

$E\left[\nabla_p f(\mathbf{x}; p_i)^2\right] =$ The moving average of the squared gradient
$E\left[\nabla_p f(\mathbf{x}; p_{i-1})^2\right] =$ The previous moving average of the squared gradient
$\gamma =$ The moving average parameter, usually 0.9

The last parameter specified in Keras when compiling a model is the metrics. When creating a neural network, a user has the opportunity to specify any parameter other than the loss value to evaluate the model during training by including it in the metrics option. For this CNN, the only other metric used to evaluate models is the performance accuracy. The performance accuracy, as was seen in the architecture optimization, is the percentage of data points which the model predicted accurately.

6.1.2. Training

After the final architecture for the CNN was defined, final training of the networks on the datasets was accomplished. Although the CNN was trained and tested on data in the previous subsection while optimizing the architecture, this section details the training

of the final four models and the hyperparameter optimization such as batch size, epochs, dropout rate, and learning rate.

Each of the four CNNs were trained on 10,000 samples from their respective datasets, validated on 2,100 samples, and tested on 2,100 samples. Since deep learning neural networks increase their performance when exposed to more data, 10,000 samples were chosen as sufficient for training. During early stages of neural network architecture optimization, fewer sample were explored to save on computational resources, but performance decreased as the number of training samples available to the network decreased. For the validation and test data subsets, 2,100 samples were chosen for each to maintain a ratio of 70% training samples and 30% validation/test samples. Table 42 shows the tensor input shape for each data subset. The values for the first dimension in each data subset refers to the number of datapoints or "images". When the training dataset is goes through the neural network, the dimension with value 10,000 is added to the output of each layer in Table 41. Table 43 shows the tensor output shapes of the models. Each model outputs *N* numbers for each inputted data point. As previously mentioned, during training, a neural network learns from a training data subset, while a validation data subset is used to evaluate the model's performance on new data during training. The training and validation loss and accuracy performance during training for the final models are seen in 6.1.2.2-6.1.2.5. Training loss and accuracy graphs are commonly used to evaluate the degree of overfitting in a model and the smoothness of training. If the training and validation curves do not converge in either graph, overfitting is present. A third subset of the data, the test dataset, is used once a model is finished training to evaluate its final

performance on new data. The test accuracies were the metric used when optimizing the CNN architecture and hyperparameters when choosing final models. The final accuracies and evaluations of the four CNNs is seen in the Results and Discussion subsection. As was the case in the architecture optimization, callbacks were used during training of the final models in order to save the best performing model during testing. Also, the CNN for each of the four datasets was trained three different times on the same data subsets and the best performing model of the three was chosen as the final model for each dataset.

Table 42. The input tensor shapes for the three data subsets used while training the CNNs.

Subset	Input Shape
Training Data	(10000, 100, 100, 1)
Validation Data	(2100, 100, 100, 1)
Test Data	(2100, 100, 100, 1)

Table 43. The output tensor shapes for the outputted data subsets for each dataset.

Subset	Dimension Dataset Output Shape	Temperature Dataset Output Shape	Gas Dataset Output Shape	Aerosol Dataset Output Shape
Training Data	(10000, 7)	(10000, 6)	(10000, 5)	(10000, 2)
Test Data	(2100, 7)	(2100, 6)	(2100, 5)	(2100, 2)

When training neural networks, it is generally good practice to normalize the input data before training [121]. Introducing data points with very large integers, like in the surrogate datasets, which are larger than the network's weight values, can cause large gradient updates which prevent convergence of the network [121]. Since network weight

values are initialized between 0 and 1, normalizing the data points to values within the same range aids in the performance of the network. For example, since the Dimensions data set has values between 100 and 189 for each pixel, all number would be normalized to between 0 and 1. Eq. 6-10 was used to normalize the samples in each dataset before going through the network for some training sessions. Training the network on normalized data increased the test accuracy scores of the model by about 0.5-1% and displayed smoother conversion in the training and validation curves. It is important to remember that when testing the model on new data, the new data must also be normalized before the model can evaluate it to make its predictions. When the models trained on normalized data were used to make predictions on new data, the model's performance decreased significantly when compared to a model trained on unnormalized input data. Although the normalized models displayed slightly higher test accuracy during training, their performance on predicting individual phenomena was surprisingly poor when using the metrics outlined in the Results and Discussion subsection. For example, the rate for predicting the presence of automobiles in the Dimension dataset correctly was around 85% for the normalized models and around 93% for the unnormalized models. Since model performance on individual phenomena is of most importance for the characterization algorithm, the final CNN architecture was trained on unnormalized input data.

$$normalized\ \mathbf{x} = \frac{\mathbf{x} - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})} \tag{6-10}$$

Where,

$\mathbf{x}$ = The input dataset tensor undergoing normalization

6.1.2.1. Hyperparameter Tuning

The hyperparameters in a network pertain to how a network is trained. Depending on the chosen values, a network can converge faster or slower, not converge at all, and even increase or reduce overfitting. A list of the different hyperparameters can be seen in Table 44, along with their different values used during optimization. As was the case during the optimization of the network architecture, the hyperparameter optimization was performed using the Dimension dataset. After a complete final architecture, including hyperparameter values, was chosen, it was then trained on the remaining three datasets to produce the other final models. The figures for the hyperparameter optimization also display a 0.26% accuracy error in training.

Table 44. A list of the tuned hyperparameters and their explored values.

Hyperparameter	Value
Learning Rate	0.0001, 0.00005, 0.00001
Epochs	20, 25, 30, 35, 40
Batch Size	2, 8, 16, 32
Dropout Rate	0.1, 0.2, 0.3, 0.4, 0.5

The first hyperparameter that was optimized was the learning rate for the network's optimizer function. During initial optimization of the network architecture and other hyperparameters, a default learning rate of 0.0001 was used for the RMSprop optimizer. While using the default learning rate for the network, a well performing CNN architecture was found and optimized. It appeared that a final CNN model was found, but there was still one issue: spiking was very prevalent in the training and validation curves. As

mentioned, even though the presence of spikes indicates that a model did not converge smoothly, models can still converge and produce high test performance. Spiking becomes a problem when its presence causes a network's final test accuracy to oscillate significantly with each training session. Training a neural network with a smaller learning rate aids in convergence by providing smaller step sizes during the gradient descent process. Smaller steps sizes ensure a gentler approach to the minima in gradient descent, while larger step sizes can cause the gradient to skip around, therefore causing spikes or preventing convergence all together. A step size too small, however, slows down the learning process for a neural network to a point where performance does not improve. In an attempt to smooth over the validation accuracy and loss curves during training, a learning rate of 0.00001 was used (one magnitude lower than the default). Although the validation and training data subset values seemed to match during the entirety of training, the new learning rate was too small to allow for the neural network to learn. The training and validation accuracies did not improve with each subsequent epoch. A third learning rate with a value halfway in between the first two was then explored. The learning rate of 0.00005 was large enough to allow the neural network to learn. Spiking in the training and validation curves was significantly reduced while maximum test accuracy only decreased by half a percent. Although performance technically suffered slightly with a training rate of 0.00005, the network displayed a lot smoother convergence during training. As a result, the final learning rate for the network was chosen as 0.00005 to increase model robustness during training. After a new learning rate was set, CNN architecture optimization was conducted again using the new hyperparameter value to validate the optimized

architecture. The optimization process described in the previous subsection details the results of the final optimization process for the CNN.

After the learning rate was set and the CNN architecture re-optimized, the rest of the hyperparameters were chosen. As was mentioned, the number of training epochs for a model refers to the number of cycles the training data is passed through the network. With each epoch, the network keeps updating its weights until it converges on a final set of parameters to reach its maximum possible performance. A larger number of epochs for a network to train on gives more time for its training and validation values to converge to a final solution, but too many epochs can cause overfitting to start occurring for certain networks. Also, too few epochs prevent the model from converging to its maximum performance. Figure 60 displays the best final accuracies produced when using each training epoch option explored, as listed in Table 44. As can be seen, the final test accuracies for all epochs are all within each other's error bars. While training the different models, the "early stopping" callback was used with a patience of six epochs. Early stopping halts the training of a model if the validation accuracy did not improve after six epochs. Due to early stopping, the models trained for longer than 25 epochs were actually halted between 25-30 epochs. Essentially, training the model longer than 30 epochs did not increase performance and created overfitting. When several models were trained using 30 total epochs, some were stopped early (while still passing the 25-epoch limit) and others reached maximum performance at 30 epochs. Since setting the number of epochs to 30 appears as the natural limit for the network, it was the value chosen for the final CNN. Although the final network is set to train over 30 epochs, the best performing model

produced during training might not occur at the end of the 30. Using early stopping permits saving the best model if it occurs at an earlier epoch.

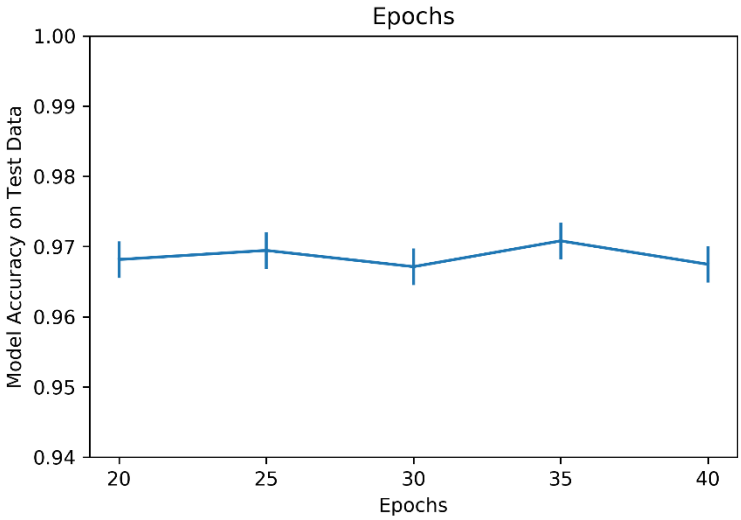


Figure 60. The final model accuracy compared to number of epochs.

The batch size while training a model refers to the number of data points (or "images") passed through the model at a time. During an epoch, the training dataset is passed through the network in intervals of data, called batches, until all the data has gone through. The gradient descent algorithms for a neural network operate with these batch sizes to update the network parameters. A larger batch size can train a network faster but may prevent the model from converging. A smaller batch size may lead to better convergence but slows down training. Four different batch sizes for the CNN were explored: 2, 8, 16, and 32. While training the models, the number of epochs were kept at 30. Figure 61 shows final accuracies depending on batch size. As can be seen, due to the network's learning rate, batch size did not affect a network's test accuracy performance.

When a higher learning rate was used in previous network iterations, it was observed that batch size did have an effect in that case. The models with 2, 8, and 16, batch sizes produced similar spiking and overfitting trends, but the model with a batch size of 32 displayed slightly less convergence during training. While the three smallest batch sizes only produced a few spikes in the validation curves, the 32-batch size model caused larger spikes and overfitting towards the later epochs. Since no difference in performance was observed between the batch sizes of 2, 8, and 16, a size of 16 was chosen to decrease the computer time for training.

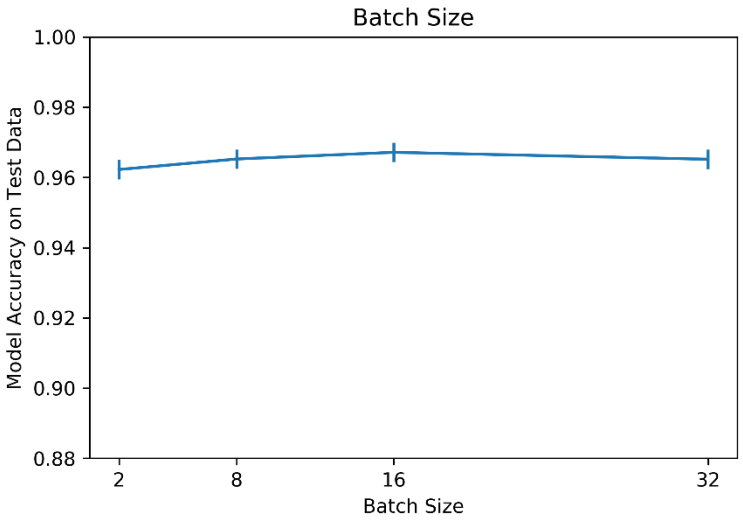


Figure 61. The final model accuracy compared to batch size.

As mentioned earlier, adding dropout to a neural network helps in reducing overfitting during training by preventing the network from learning irrelevant features from the dataset that can hinder performance. When adding dropout to a layer in a network, the dropout rate must be specified. Typically, dropout rates between 0.2-0.5 are used

[121]. Higher dropout rates are better at reducing overfitting while training, but if a rate is too large, the network may not learn. Figure 62 shows the final model accuracies for the different dropout rates listed in Table 44. A. slight trend where performance increases is present. The two highest rates produced the best performance and are outside of the error for the lowest dropout rate. Although test accuracy was similar with the three highest rates, a significant trend in the training curves was observed where overfitting decreased in the validation data as dropout rate increased. Since a dropout rate of 0.5 produced one of the highest test accuracies and the least overfitting of the data, it was chosen as the final rate for the network.

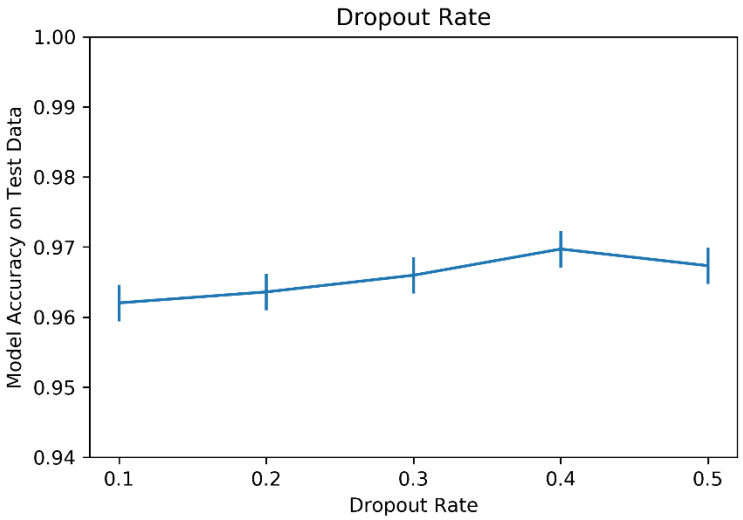


Figure 62. The final model accuracy compared to dropout rate.

In summary, Table 45 lists the final hyperparameter values used while training the final CNN architecture. The figures in 6.1.2.2-6.1.2.5 show the training and validation data subset accuracy and loss values produced while training the final CNN architecture

on each dataset. While training, each CNN model was run a total of three different times and the "save the best model" callback was used during each training session to prevent spiking or overfitting to hinder the final model performance. In the figures below, a red line marks the location of the epoch at which the best model was saved during training. For each of the four CNNs, the best performing model out of the three training sessions was chosen as the final model. Although the validation curves in the final model figures display spiking, it does not mean the models perform worse than if it had a smooth curve. Each model still produced high performance accuracies that perform with consistency, as seen in the Results and Discussions subsection, even though their training convergence was not the smoothest.

Table 45. The final hyperparameter values chosen.

Hyperparameter	Value
Learning Rate	0.00005
Epochs	30
Batch Size	16
Dropout Rate	0.5

6.1.2.2. Dimension Dataset Final CNN Training Performance

Figures 63 and 64 below show the training and validation accuracy and loss values for each epoch during training for the Dimension dataset final CNN model. The red line indicates at which epoch the best performing model was saved. In this case, the best performing model occurred at epoch 28 since the model began to overfit after the 28th epoch. As can be seen, there were a few peaks in the validation accuracy and loss values

before training was halted. The presence of spikes does not indicate the presence of overfitting or hinder the model's final performance. As was mentioned earlier, when the CNN was trained on normalized data, the frequency of spiking was reduced and overfitting towards later epochs eliminated. Figures 65 and 66 show the training and validation curves for the Dimension CNN trained on normalized data. One thing to note for the normalized model is that it the total number of training epochs were optimized to 20 instead of 30. Although normalized models displayed better training performance, they did not perform well at all on the metrics described in the Results and Discussion subsection.

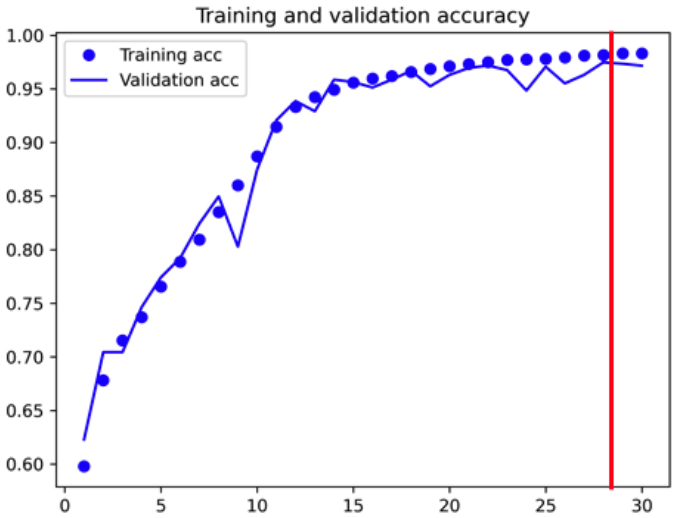


Figure 63. The training and validation data subset accuracy during training for the Dimension CNN model.

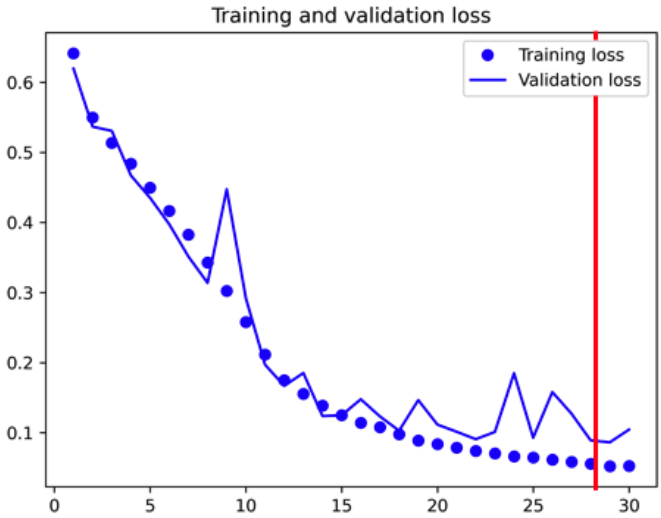


Figure 64. The training and validation data subset loss during training for the Dimension CNN model.

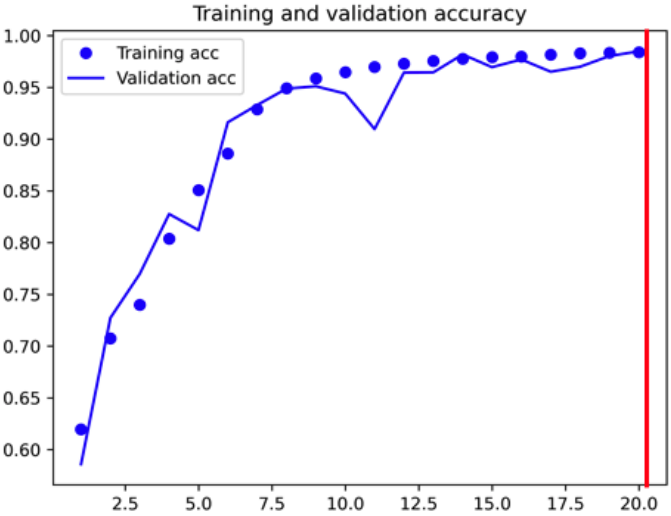


Figure 65. The training and validation data subset accuracy during training for the Dimension CNN model with normalized input data.

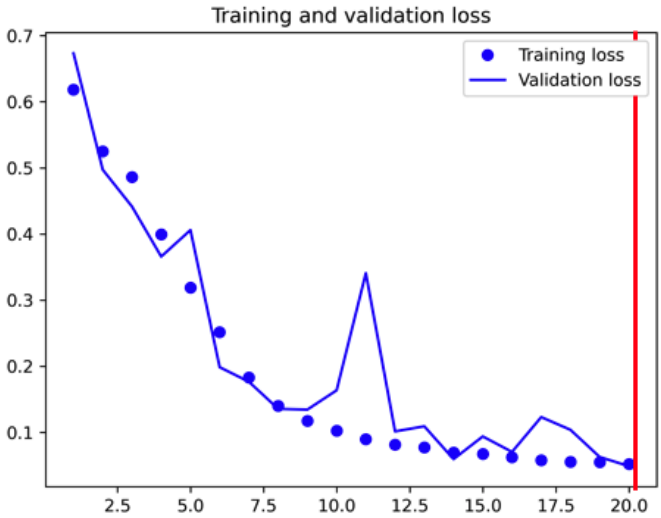


Figure 66. The training and validation data subset loss during training for the Dimension CNN model with normalized input data.

6.1.2.3. Temperature Dataset Final CNN Training Performance

The figures below show the training and validation accuracy and loss values for each epoch during training for the Temperature dataset final CNN model. The red line indicates at which epoch the best performing model was saved. In this case, the best performing model occurred at epoch 17 since the model started overfitting after it. It is interesting to observe that the CNN model trained on the Temperature dataset was halted seven epochs the Dimension CNN was and at almost halfway of the 30 training epochs it was set to. Without the "save the best model" option during training in Keras, the Temperature CNN would have continued to train causing increased overfitting as the number of epochs reached 30, resulting in terrible model performance. As can be seen, before the best model was saved at epoch 17, the model suffered a couple of big spikes during training. The presence of spikes does not indicate the presence of overfitting or hinder the model's final performance. Like with the Dimension CNN, the presence of

overfitting towards later epochs was eliminated when using normalized data, but the unnormalized model performed a lot better in the metrics described in the Results and Discussion subsection.

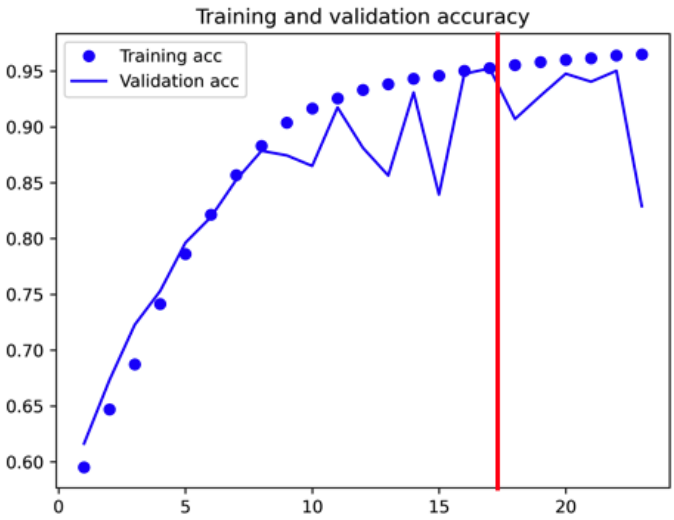


Figure 67. The training and validation data subset accuracy during training for the Temperature CNN model.

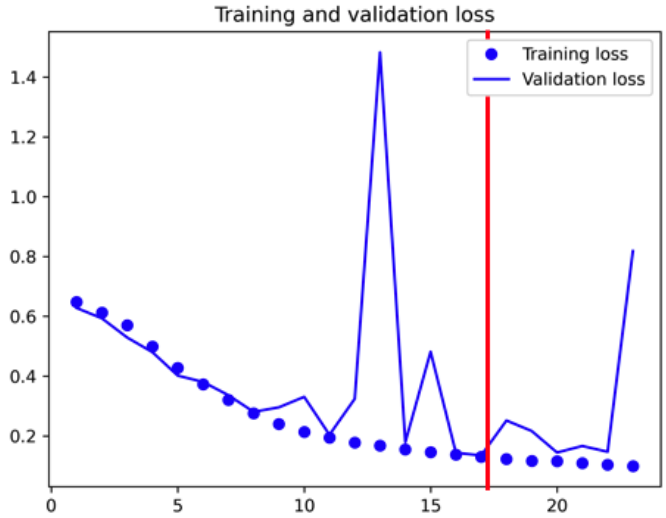


Figure 68. The training and validation data subset loss during training for the Temperature CNN model.

6.1.2.4. Gas Dataset Final CNN Training Performance

The figures below show the training and validation accuracy and loss values for each epoch during training for the Gas dataset final CNN model. The red line indicates at which epoch the best performing model was saved. In this case, the best performing model occurred at epoch 20 since overfitting began occurring after it. As can be seen, convergence for the gas model was a lot smoother compared to the previous two models. However, overfitting started occurring in the model after a certain number of epochs, like in the other models, preventing it from completing the 30 total epochs. It is interesting to note that the Gas CNN was halted between the epochs where the previous two models were halted. The difference in early stopping originates from the use of unnormalized data. As was illustrated for the Dimension CNN, normalized input data smoothened the model's convergence, reduced the presence of spiking, and eliminated overfitting. When training each of the four CNNs on normalized data, their best model epochs were all at the final epoch or a few epochs before. Using unnormalized input data, on the other hand, creates more volatility when training because of increased spiking and overfitting, leading to large differences in early stopping epochs between the four CNNs.

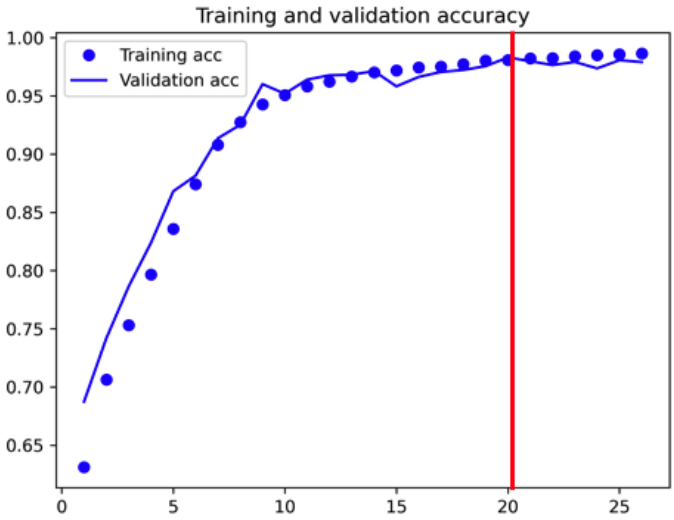


Figure 69. The training and validation data subset accuracy during training for the Gas CNN model.

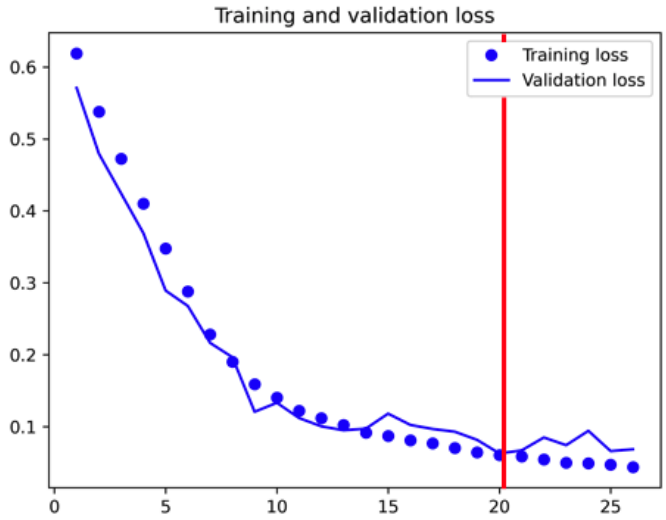


Figure 70. The training and validation data subset loss during training for the Gas CNN model.

6.1.2.5. Aerosol Dataset Final CNN Training Performance

The figures below show the training and validation accuracy and loss values for each epoch during training for the Aerosol dataset final CNN model. The red line indicates at which epoch the best performing model was saved. In this case, the best performing model occurred at epoch 17. Even though the presence of overfitting in the last epochs is not clear in the graphs, training of the model was halted because performance did not improve within six epochs after the best model was saved.

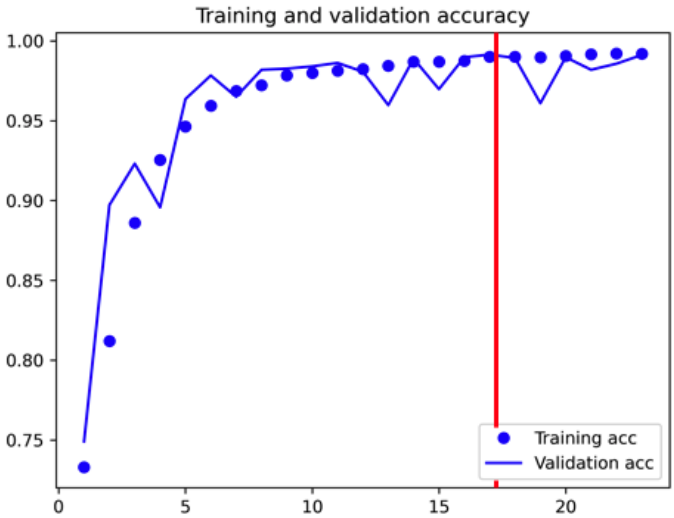


Figure 71. The training and validation data subset accuracy during training for the Aerosol CNN model.

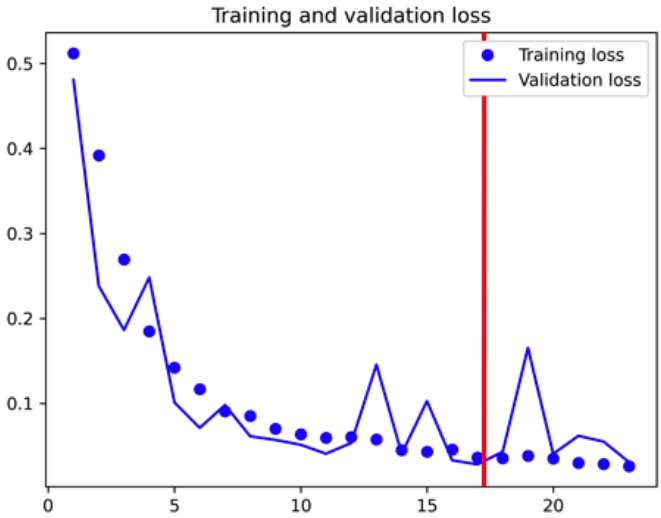


Figure 72. The training and validation data subset loss during training for the Aerosol CNN model.

6.2. Results and Discussion

While the previous subsection discussed the characterization methodology's CNN architecture and training, the results of each CNN and the overall methodology are detailed in this subsection. During optimization and training of the CNNs, the models' test accuracy was used to compare their performance. As mentioned, the test accuracy refers to the percentage of labels correctly predicted for a new sample dataset. Table 46 lists the final model accuracies, loss scores, and their performance errors for each CNN. While the previous subsection referred to training error as the difference in test accuracy produced each time a network architecture was trained, the performance error of a model is the difference in accuracy one single model experiences every time it evaluates new data. As seen in Table 46, the final Dimension CNN model selected produced a test accuracy of around 97%. Once a model is trained, it can then be used to evaluate new data at any point in the future. The error for the Dimension CNN was produced by calculating the standard

deviation in accuracies the CNN outputted every time it evaluated a new sample of data. For each evaluation of all four models, a test data subset of size (2100, 100, 100, 1) was used for consistency. Each CNN was evaluated on ten different test datasets, and Table 46 shows the mean and standard deviation for accuracies and loss scores produced from the ten evaluations.

Table 46. The final accuracies and loss scores for each CNN model in the characterization methodology.

Model	Test Accuracy	Test Loss
Dimension	97.318% ± 0.159%	0.0933 ± 0.0059
Temperature	95.272% ± 0.106%	0.1375 ± 0.0044
Gas	98.263% ± 0.098%	0.0621 ± 0.0028
Aerosol	99.038% ± 0.162%	0.0337 ± 0.0044

As can be seen in Table 46, all four CNN models produced a test accuracy higher than 95%. When comparing test accuracies between the different models, a trend appears where test accuracy increases as the number of labels in the dataset decreases. As the number of labels to predict for data points decrease, the prediction problem for a neural network becomes easier to solve since it needs to learn less relationships in the data. Interestingly, the Temperature CNN model does not follow this trend. Since the Temperature dataset includes one less label per "image" than the Dimension dataset, a test accuracy equal to or higher than the Dimension CNN was expected. CNN architecture and hyperparameter optimization was performed on the Temperature CNN to find an alternative configuration that performed better on the Temperature dataset. Different convolutional and dense layers, different filter and neuron numbers, higher and lower

batch normalization, different dropout rates, and different epoch and batch sizes were all explored, but none of the alternate architectures performed better than the final CNN model previously created. The source of the decrease in performance of the Temperature model is most likely due to the nature of the "images" in the dataset rather than the CNN architecture. Since three of the six objects included in the Temperature dataset used the same number of pixels, the CNN architecture was not able to learn enough representations in the data to distinguish the phenomena and produce a higher performance than the Dimension CNN.

Although test accuracy is a useful metric to compare different CNN architecture performances during training, it is not a good indicator to how well the models predict the presence or absence of individual phenomena within an image. Instead, the following metrics are used for gauging a model's performance on each phenomenon: the True Positive (or Recall Rate), the False Positive (FP) Rate, the True Negative (TN) Rate, the False Negative (FN) Rate, and the Precision Rate. The Recall Rate is the ratio of the number instances a model correctly predicted the presence of a phenomenon over the true number of instances when the phenomenon was present, as described in Eq. 6-11. The False Positive Rate is the number of times the model predicted a phenomenon was present but was incorrect. The True Negative Rate is the number of instances a model correctly predicted the absence of a phenomenon. The False Negative Rate is the number of instances a model predicted the absence of a phenomenon but was incorrect. Finally, the Precision Rate is the ratio of the number of instances the model correctly predicted the presence of a phenomenon over the total number of times the model predicted the presence

of a phenomenon, as described in Eq. 6-12. When analyzing the performance metrics, only the Recall and Precision Rates are of relevance since the other rates can be deduced from these two. Tables 47-50 show the performance metrics for the four different models. When calculating the performance rates, each model evaluated 5000 total samples for its respective dataset. As was mentioned in the Training subsection, the metrics in Tables 47-50 are where the CNN models trained on normalized data did not perform well. For example, the recall rate on automobiles for the Dimension CNN was around 92%, while the recall rate in the same category for the Dimension CNN trained on normalized data was around 85%. The decrease in performance for phenomena identification with the normalized CNNs is most likely due to the loss of information about the pixel values in the "images" by normalizing them.

$$RR = \frac{TP}{GP} \tag{6-11}$$

Where,

$RR =$ Recall Rate

$TP =$ Number of instances a phenomenon was correctly predicted present

$GP =$ Total number of instances a phenomenon is present

$$PR = \frac{TP}{TP+FP} \tag{6-12}$$

Where,

$PR =$ Precision Rate

$FP =$ Number of instances a phenomenon was incorrectly predicted present

Table 47. The performance of the Dimension CNN for identifying phenomena of interest.

Phenomena	Recall Rate	Precision Rate	FP Rate	TN Rate	FN Rate
Automobile	92.5%	99.9%	0.08%	99.9%	7.47%
Airplane	96.1%	99.2%	0.77%	99.2%	3.93%
Facility	98.0%	98.5%	1.43%	98.6%	1.97%
Cons/Mine Vehicle	96.0%	99.8%	0.16%	99.8%	4.01%
Footprint	99.6%	99.9%	0.08%	99.9%	0.44%
Fire	93.7%	97.4%	2.54%	97.5%	6.34%
Blackout	95.4%	98.0%	2.03%	98.0%	4.58%

Table 48. The performance of the Temperature CNN for identifying phenomena of interest.

Phenomena	Recall Rate	Precision Rate	FP Rate	TN Rate	FN Rate
Automobile	90.0%	99.8%	0.16%	99.8%	10.0%
Airplane	94.1%	99.5%	0.43%	99.6%	5.86%
Fire	93.9%	96.8%	2.85%	97.1%	6.06%
Blackout	90.6%	92.3%	7.95%	92.1%	9.41%
Cons/Mine Vehicle	97.2%	99.3%	0.73%	99.3%	2.77%
Cons/Mine Process	89.7%	99.8%	0.20%	99.8%	10.3%

Table 49. The performance of the Gas CNN for identifying phenomena of interest.

Phenomena	Recall Rate	Precision Rate	FP Rate	TN Rate	FN Rate
Automobile	95.6%	99.7%	0.33%	99.7%	4.35%
Airplane	97.4%	98.8%	1.23%	98.8%	2.58%
Fire	95.0%	99.9%	0.12%	99.9%	4.95%
Cons/Mine Vehicle	97.1%	99.5%	0.52%	99.5%	2.92%
Cons/Mine Process	99.1%	99.6%	0.41%	99.6%	0.91%

Table 50. The performance of the Aerosol CNN for identifying phenomena of interest.

Phenomena	Recall Rate	Precision Rate	FP Rate	TN Rate	FN Rate
Fire	96.9%	99.8%	0.16%	99.8%	3.08%
Cons/Mine Process	99.3%	99.9%	0.12%	99.9%	0.74%

It can be seen in Tables 47-50 that the test accuracies produced by the CNNs during training do not necessarily translate to their performance in predicting the presence of phenomena. However, CNNs with lower test accuracies produced lower Recall Rates than CNNs with higher test accuracies. As can be seen, the rates listed in Tables 47-50 demonstrate the high effectiveness of the CNN models. Out of all the Recall Rates, only seven were below 95%, and out of those seven, only one was below 90%. To reiterate, the Recall Rate refers to the probability a CNN will predict the presence of a phenomenon when it is actually there. In the Dimension CNN, only the prediction of automobiles and fires is below 95%. About 7.47% and 6.34% of the time, the Dimension CNN will not recognize the presence of automobiles and fires, respectively. In the Temperature CNN, only one phenomenon has a Recall Rate higher than 95%. The lowest performing phenomena are automobiles and construction/mining processes. For the Gas and Aerosol CNNs, all phenomena Recall Rates perform higher than 95%. As a gauge of performance, other CNNs trained on satellite imagery for object identification produced Recall Rates of 87-95% [125]. Although the CNNs used for comparison were trained on real satellite images and were detecting different phenomena, their Recall Rates serve a threshold for typical CNN object detection performance. As for Precision Rate, most phenomena produced rates higher than 99%, with only two phenomena producing rates lower than

97%. The two least-performing phenomena for Precision Rate are fires and blackouts in the Temperature CNN. High performance in Precision Rates indicate that False Positives are not an issue for the characterization methodology. For comparison, typical Precision Rates for other object detection CNNs are 82-97% [125].

When analyzing all the results, the only noteworthy issues in the CNNs are the >9% False Negative Rate for automobiles, blackouts, and construction/mining processes in the Temperature CNN. About one out of ten times, the Temperature CNN will not recognize the presence of those three phenomena. Fortunately, phenomena probabilities from each CNN are combined in the last step of the characterization methodology. As seen in Figure 52, using the four CNNs to predict the presence of phenomena in an image is not the only aspect of the overall characterization methodology. Eq. 6-1 is used to average the probabilities for each phenomenon to produce a final probability. The best way to illustrate the last step of the methodology is through an example.

A target of interest for surveillance includes 4 automobiles, 2 facilities, and 1 fire. The characterization algorithm receives four images containing the phenomena, one for each sensor type (dataset). In the Dimension dataset image, all seven objects are visible. In the Temperature and Gas dataset images, only the automobiles and the fire are visible. Finally, in the Aerosol dataset image, only the fire is visible. After passing the four images through their respective CNNs, predictions for the phenomena's presence were made. Table 51-54 display the prediction outputs for each CNN.

Table 51. The phenomena probability predictions from the Dimension CNN.

	Automobile	Airplane	Facility	C/M V	Footprint	Fire	Blackout
Probability	1	0.008	0.999	0.019	0.001	0.991	0.006

Table 52. The phenomena probability predictions from the Temperature CNN.

	Automobile	Airplane	Fire	Blackout	C/M V	C/M P
Probability	0.999	0.032	0.974	0.048	0.037	0.025

Table 53. The phenomena probability predictions from the Gas CNN.

	Automobile	Airplane	Fire	C/M V	C/M P
Probability	1	0.014	0.933	0.016	0.076

Table 54. The phenomena probability predictions from the Aerosol CNN.

	Fire	C/M P
Probability	0.999	0.001

As can be seen in the tables above, each individual CNN outputted the probability of every phenomena possible for the dataset. Each CNN was able to correctly predict the presence of the phenomena in each image. Once the probabilities are produced, Eq. 6-1 is used to average the probabilities of each phenomenon. As mentioned, a probability is only incorporated into Eq. 6-1 if it exists. For example, since automobiles are not visible in the Aerosol dataset, only its probabilities from the other three CNNs are averaged. Table 55 shows the final characterization methodology solution for the example scenario. As can be seen, the methodology outputted a probability higher than 97% for the phenomena actually present in the image, indicating that characterization was successful.

Table 55. The phenomena probability predictions from the Dimension CNN.

	Automobile	**Airplane**	**Facility**	**C/M V**
Probability	0.999	0.018	0.999	0.024
	C/M P	**Footprint**	**Fire**	**Blackout**
Probability	0.034	0.001	0.974	0.027

The results in Table 55 exhibit how Eq. 6-1 takes advantage of the increased characterization accuracy and robustness that comes from the use of multiple sensors onboard the CubeSat system. If one sensor malfunctions or if one CNN outputs an incorrect probability, the last step of the characterization methodology can still make an accurate prediction about the target of interest. For example, if the Gas CNN outputted a probability of 30% for the fire in the previous example, the characterization methodology would still output a high final probability of 81.6% since it takes advantage of the other CNNs' correct predictions. Even if the Aerosol CNN also outputted a low probability of 30%, the final probability would output 64.1%, which is better than a 50/50 guess. However, if three out of the four CNNs malfunction, the final prediction for a phenomenon would be incorrect. Therefore, it is extremely important to adequately train the CNNs to output high accuracies. Also, if a phenomenon has a low Recall Rate for one CNN but a very high rate in another, its final probability of detection is not affected (like construction/mining processes).

6.3. Limitations

As seen above, the characterization methodology is very effective in predicting the presence of phenomena of interest in a given seen. The four CNNs used for the different

sensor types produced recall rates and precision rates above 90% for all phenomena. Although effective for the data and problem posed in this thesis, the characterization methodology has a few limitations. The methodology's drawbacks are caused by the type of data used to train the CNNs in this thesis and by inherent limitations of using convolutional neural networks.

As was stated previously, the surrogate dataset created in Section 5 aimed at creating "images" that were simple representations of actual sensor data. Consequently, the CNN models trained in this thesis can only predict labels for "images" created with the surrogate dataset methodology. The CNN models, as they currently stand, cannot make any predictions on data or images collected by sensors onboard a CubeSat. They simply were not trained to do so. The beauty of creating a characterization methodology on simple datasets, however, is that the architectures (CNNs or other aspects of the methodology) present a general basis for prediction algorithms that can later expand to accommodate more complex datasets. It is possible for the characterization methodology architecture, as it stands, to successfully predict the presence of the phenomena of interest in actual satellite images given adequate training data. As the characterization methodology is further developed in the future for a CubeSat platform, its architecture can be easily altered to accommodate new challenges (if adequate data is available).

The necessity of adequate data for training creates the limitations inherent to any algorithm involving convolutional neural networks. CNNs not only require vast amounts of data to boost their performance, but the data must also be adequately labeled. Since CNNs fall under the Supervised Learning umbrella of deep learning, they are only

effective if they know what they should predict. Although multitudes of satellite image databases are available to users, a database of thousands of images containing the phenomena of interest to the CubeSat satellite system is required to develop an accurate characterization methodology for integration with the platform. Even then, the images in the database might not have the same resolutions as the sensors onboard the CubeSat system.

The solution to developing an accurate characterization methodology for true CubeSat images is to launch an exploratory CubeSat mission with identical satellite architectures and sensors with the goal of collecting as many images of phenomena of interest as possible. The first CubeSat system mission would only serve the purpose of data collection. The collected images can then be used to train the characterization methodology for a second CubeSat mission.

6.4. Illustrative Application

To summarize all the topics covered in this thesis thus far, an illustrative application of the CubeSat surveillance system is explored. The following example explains how a final functioning CubeSat surveillance system would operate to autonomously characterize phenomena of interest in a target location. To reiterate, the overall CubeSat data and algorithm interface introduced in Section 3 is seen in Figure 73.

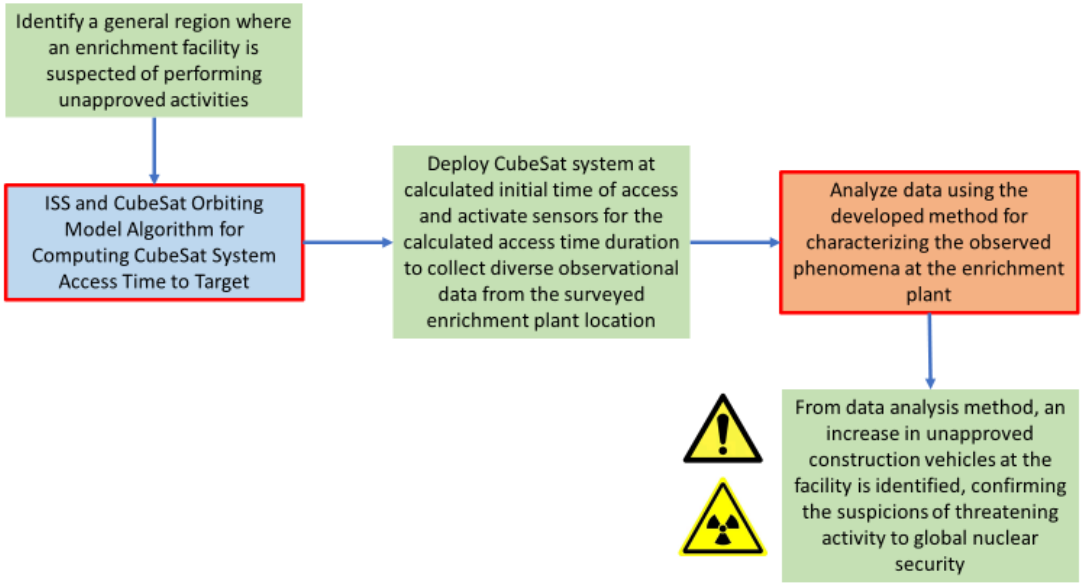


Figure 73. The data flow for observing and identifying suspicious activity at an enrichment plant.

To initialize the example, it is assumed that a nine-CubeSat constellation is stashed within a deployer attached to the ISS. All of the satellites within the constellation have the same component architecture except for their payloads. Two CubeSats in the system each feature a combination of panchromatic and multispectral sensors which capture images within the visible and near-infrared (VIS-NIR) electromagnetic spectrum. Another two CubeSats each feature a multispectral sensor which operates within the infrared (IR) spectrum. Another two each feature a hyperspectral sensor which operates in the IR spectrum. Another two each feature a multispectral sensor which operates in the ultraviolet (UV) spectrum. Finally, the ninth CubeSat features a large computational component as its payload where the images collected by the other eight satellites are processed by the characterization algorithm. Each CubeSat also contains S-band receivers and X-band

transmitters programmed to effectively communicate with the Globalstar satellite relay network and KSAT ground station network. The Globalstar relay allows the CubeSats in the constellation to communicate with each other, while the KSAT ground stations allow the CubeSats to downlink data to the surface.

On November 13th, 2020, word is received at the CubeSat system command center that there are rumors of suspicious activity at an enrichment facility near Tehran, Iran. The US Government wishes to use the CubeSat surveillance system to image the target location and determine if there are any phenomena of interest which indicate nefarious activity. Since the CubeSat surveillance system is currently stashed within the ISS, the deployment algorithm is initialized to calculate the deployment time for the CubeSats. First, the ISS ephemeris data for November 13th was used to model its orbit within NASA's GMAT. Given Tehran's longitudinal coordinates of 35.6892° N, 51.3890° E, an altitude of 1189 m, and an ISS angle of elevation of 7°, the deployment algorithm and GMAT determined that the ISS first has access to Tehran at 16:17:56.070 UTC on November 13th. The complete ISS access time is seen in Table 56. The ISS's first access time only lasting 77.59 s means its orbit most passed through the fringes of the field of view. Since the ISS has a smaller angle of elevation than the CubeSat sensors, the CubeSat system will most likely not have access to Tehran, or at least an access time with a significant duration. However, there is no disadvantage in deploying the CubeSat system early, so t_1 for the ISS in the deployment algorithm is set to 16:17:56.070 UTC. Once t_1 of the ISS's access time was calculated, the CubeSat deployment time is defined as $t_{CS} = t_1 - 30\ min$. By deploying the CubeSat system 30 minutes before their orbit passes over

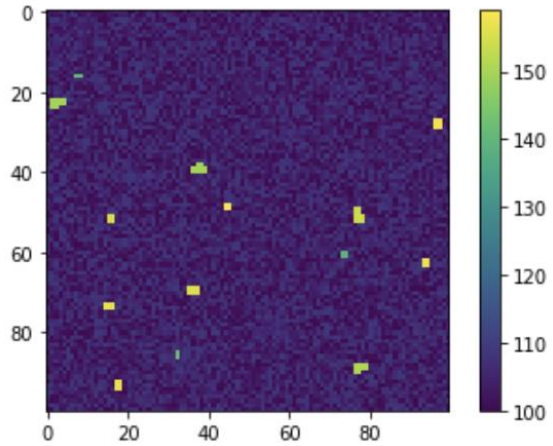
Tehran allows enough time for the satellites to boot up. Since the CubeSat system will most likely not have significant access to Tehran on the first fly-by, satellite bootup speeds are not a concern in this scenario. The CubeSats are deployed at 15:47:56.070 UTC. To model the CubeSat system orbit within GMAT, the ephemeris data from the ISS at 15:47:56.070 is set as the CubeSats' initial ephemeris. Given Tehran's coordinates and an angle of elevation for the CubeSats of 60°, the CubeSats pass over the target at 17:54:19.509, as seen in Table 56. As expected, the CubeSats had significant access to Tehran during the ISS's second pass. Since the CubeSats were deployed two hours before they reached the target, the satellite systems were boot up completely and good to go. As seen in Table 56, the sensors have over a maximum of 62 s to image the enrichment facility near Tehran. Although the CubeSat system passes over Tehran for a longer period of time, the sensors' angle of elevation limit true access to only about a minute.

Table 56. The satellite access times.

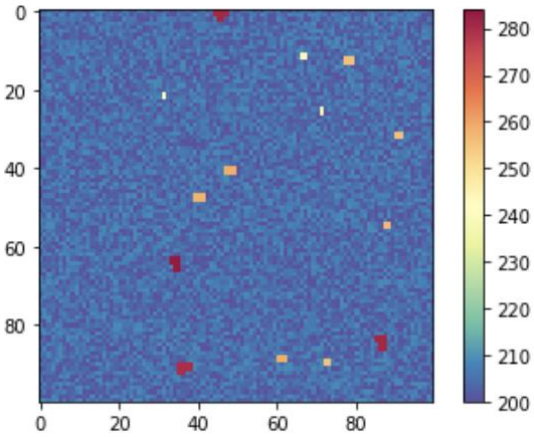
Satellite	Access Time Start Time (UTC)	Access Time End Time (UTC)	Duration (s)
ISS	16:17:56.070	16:19:13.658	77.59
	17:51:02.995	17:58:40.185	457.2
CubeSat System	17:54:19.509	17:55:22.355	62.85

As the CubeSat system passes over the enrichment facility near Tehran, its sensors collect data from the scene. The images captured by each sensor type are then transmitted to the CubeSat with the larger computational unit to analyze the images. Figure 74 shows the images captured by the panchromatic/multispectral sensors in VIS-NIR, the multispectral sensors in IR, the hyperspectral sensors in IR, and the multispectral sensors

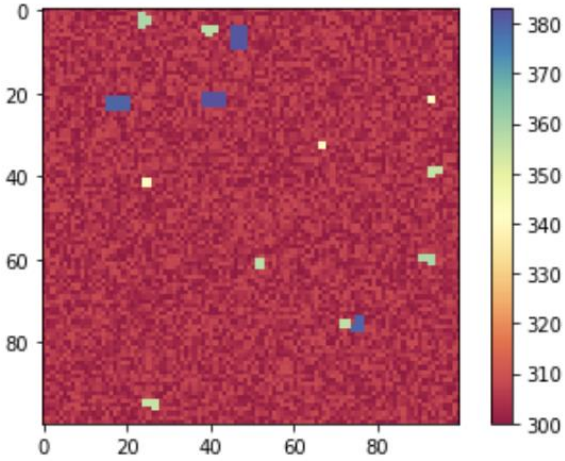
in UV, respectively. Figure 74 only shows the objects visible to each dataset. As a reference, the scene of interest includes 4 facilities, 3 automobiles, 7 construction vehicles, and 4 construction processes. It is important to note that the objects in the images are randomly distributed due to the nature of how they are created in the datasets, so they do not reflect the true coordinates of the objects as for the sake of the example. Also, false objects are included in the images to present a challenge for the characterization methodology.



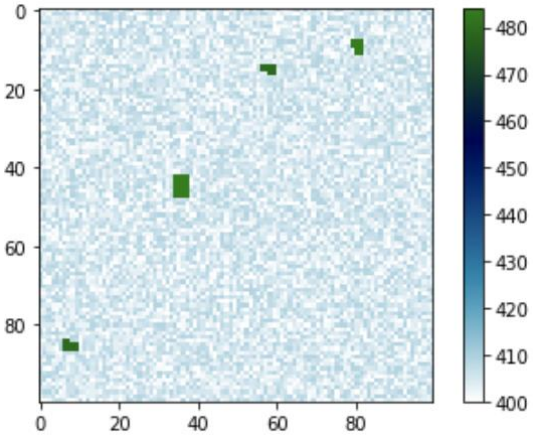
The image captured by the panchromatic/multispectral sensors in VIS-NIR.



The image captured by the multispectral sensors in IR.



The image captured by the hyperspectral sensors in IR.



The image captured by the multispectral sensors in UV.

Figure 74. The images captured by the different sensors onboard the CubeSat surveillance system.

After all the images are received at the computational CubeSat, the characterization algorithm proceeds to make prediction on the phenomena. Table 57 shows the prediction results of the four CNNs of the characterization algorithm. As can be seen, each CNN accurately predicts the presence of phenomena in the scene by outputting a probability between 0.999-1. One prediction probability did not fall within a high accuracy range,

however. The Dimension CNN outputted a probability of 0.73 for the presence of automobiles in the scene. Since it is above 50%, the presence of automobiles is likely detected by the Dimension CNN, but the accuracy is not as high as the other CNNs.

Table 57. The phenomena presence probabilities outputted by each CNN for each image and the final characterization solution.

The phenomena probability predictions for the panchromatic/multispectral sensor image in VIS-NIR.

	Automobile	Airplane	Facility	C/M V	Footprint	Fire	Blackout
Probability	0.730	0.010	1	1	0	0.003	0.021

The phenomena probability predictions for the multispectral image in IR.

	Automobile	Airplane	Fire	Blackout	C/M V	C/M P
Probability	0.999	0.003	0.022	0.063	1	0.999

The phenomena probability predictions for the hyperspectral image in IR.

	Automobile	Airplane	Fire	C/M V	C/M P
Probability	1	0.003	0.001	1	1

The phenomena probability predictions from the Aerosol CNN.

	Fire	C/M P
Probability	0.001	1

The final characterization solution for the Tehran enrichment facility.

	Automobile	**Airplane**	**Facility**	**C/M V**
Probability	0.910	0.005	1	1
	C/M P	**Footprint**	**Fire**	**Blackout**
Probability	0.999	0	0.007	0.042

Once the CNN probabilities are calculated, the characterization produces a final characterization solution, as seen in Table 57. Due to the diversity of sensors on board the CubeSat, the low automobile probability in the Dimension CNN did not affect significantly affect the final solution probability as it is still above 90%. The final

characterization solution is then downlinked from the CubeSat system to the surface at the next visible KSAT ground station. Once data is received by the KSAT station, it is sent to the CubeSat system command center where US Government officials can interpret the results. It is determined that the presence of construction vehicles and processes at the enrichment facility presents evidence of nefarious action since such activity was never approved by the IAEA or UN.

6.5. Conclusion

This section introduced the characterization methodology developed for the identification of phenomena of interest from the CubeSat surveillance platform. The characterization methodology consists of four convolutional neural networks trained for each sensor type onboard the CubeSat system. The CNNs for this thesis were trained on the four surrogate datasets representing simplified versions of satellite imagery. CNN model architecture optimization and its network functions were discussed. After hyperparameter optimization of the CNNs and training, a final model was produced for each of the four datasets with high test accuracies. Then, the performance of each CNN on actually predicting each phenomenon in an image was discussed. Within the characterization methodology, the predictions from each CNN are combined in the last step to produce a final characterization solution to be transmitted by the CubeSat system. Finally, limitations for the characterization methodology and an illustrative application were discussed.

7. MICROREACTOR APPLICATIONS

As was mentioned in Section 1, this thesis also explores the possibilities of extending the methodology developed for the CubeSat surveillance system for the monitoring microreactors. Some of the concepts developed thus far in this thesis can be either directly or indirectly applied for ensuring safety or aiding in operations for microreactors installed in remote locations on or off planet. This section serves as an introduction to future work explored beyond this thesis.

7.1. Microreactor Definitions

With the advent of multiple advanced reactor designs, microreactors present a viable solution for clean energy production in remote areas that cannot afford the implementation of other energy sources due to a lack of resources or adverse climate. As mentioned, microreactors are defined as small units producing less than 20 MWe which feature factory assembly, high transportability, and self-regulation [57]. As an example, microreactors could be transported to small communities in northern Alaska for reliable energy production. These communities are too far away from bigger population centers to economically set up power lines, and the climate is too extreme to allow for diesel generators to run uninterrupted. A microreactor could meet those communities' energy needs by producing electricity on-site for as long as the core's life lasts, which typical designs allow for up to ten years. Also, a microreactor's self-regulating, or autonomous, design allows for them to be installed in areas where personnel are not needed. This prevents exhausting human resources and allows for microreactors to be implemented in more remote locations. Apart from extreme climates, microreactors can also meet clean

energy requirements in countries lacking infrastructure for conventional reactors, they can provide emergency energy production in disaster areas, and they can even provide reliable electricity generation for space environments.

Even though microreactors present promising designs, they are still faced with obstacles preventing their final implementation. Numerous regulatory concerns arise from the desired transportability and autonomy of these reactors. While both microreactor features are what make the designs so attractive, they present significant challenges to the current nuclear industry infrastructure. For example, enough measures must be created and implemented which assure the safety and security of a microreactor core while it is transported by truck. Regarding safety, an accident on the highway could cause a release of radioactive material, so new containers must be designed that provide protection against radiation leakage and critical configurations during normal and emergency situations [130]. For security, the transportation of high assay low-enriched uranium (HALEU) falls under the physical protection requirements within 10 CFR 73, but there currently is no pathway for licensing as no facilities have yet to receive licenses for possessing HALEU, and the protective measures must be reevaluated [130]. Even if transporting a microreactor presents no issues, the safety of its autonomous operation once installed at a site must be ensured. In order to prevent any reactor accidents, sufficient measures are needed while operating the reactor to avoid the propagation of dangerous situations. For example, fuel temperatures can be monitored to adjust coolant mass flow rates accordingly.

The challenges just presented towards microreactor deployment can be mitigated, in part, by the methodologies developed in this paper in contribution to the CubeSat-based

surveillance system. By either directly using the CubeSat system or adapting some of its approaches, progress towards ensuring the safe and secure transportation and autonomous operation of the reactors can be achieved.

7.2. Direct Application of CubeSat Surveillance System Methodology

As was proven in the previous six sections of the thesis, a system of CubeSats equipped with adequate sensors and data processing techniques provides an effective remote monitoring system for phenomena of interest. Since deploying and operating microreactors in various locations pose regulatory challenges around the world, their observation falls directly under the purview of the CubeSat surveillance system. Like with its other phenomena of interest, the CubeSat constellation can be directly used to monitor the activity of microreactors.

Like it did with automobiles and airplanes, the CubeSat system can be trained to detect and characterize the vehicles utilized for transporting microreactors. For example, the microreactor's transport truck can be characterized as it drives down a highway. As is the nature of the orbital properties of a remote monitoring system in the atmosphere, however, the CubeSats cannot maintain continuous monitoring of the microreactor as it makes its journey from the factory to the installation sight. The few images the CubeSats collect as they briefly fly over the transport vehicle could still help ensure the microreactor's integrity for those brief moments, however.

Once the microreactor is installed at its final site, the CubeSat system can characterize the plant on every access time. The dimensions of the reactor and adjacent facilities can be observed with the panchromatic/multispectral sensors onboard the

surveillance system. As an example, Table 58 lists the dimensions of a two-facility microreactor operations site proposed by Ultra Safe Nuclear Corporation at the Chalk River National Lab in Canada [131]. Similar to the enrichment facility example in the last section, the detection of other phenomena of interest near the microreactor plant can also indicate to suspicious activity. Like with the other phenomena of interest, however, enough data (or images) of microreactor plants are needed to adequately train the CubeSats' characterization methodology and learn the plants' characteristic features. Also, infrastructural emergencies, like fires and blackouts, may also occur at a plant through the same capacities the CubeSat system already possesses. Unfortunately, the CubeSat surveillance system cannot increase the autonomy of operational procedures for microreactors.

Table 58. The dimensions of a two-facility microreactor plant.

	Dimension	Value
Facility 1	Length	130 m
	Width	96 m
	Maximum Height	30 m
Facility 2	Length	180 m
	Width	102 m
	Maximum Height	17 m

Although it can provide valuable information, the CubeSat surveillance system cannot fully ensure the safe transportation and operation of microreactors since its monitoring is restricted to LEO. However, some methodologies developed as part of the

surveillance system can be applied to other sensor systems around a microreactor in a useful manor.

7.3. Indirect Application of CubeSat Surveillance System Methodology

Some aspects of the CubeSat methodology have the potential of adaptability to other applications for microreactor monitoring, namely its deep learning techniques. Machine/deep learning algorithms coupled with sensor systems within and around a microreactor can be used to improve the safety of its transportation and autonomous operation. Although the CubeSat characterization methodology was limited to supervised learning techniques, a microreactor monitoring system does not possess the same limitations. Unsupervised learning techniques for machine learning have shown effectiveness in anomaly detection situations. This subsection presents the potential of deep learning techniques for microreactor deployment through the illustration the following.

7.3.1. Example: Microreactor Transportation

As mentioned, an attractive feature of microreactors is that its fresh fuel is integrated within the reactor system before it is transported to its installation site. However, current gaps in regulations currently prevent this transportation of microreactors [130]. According to the Nuclear Regulatory Commission's (NRC) 10 CFR Part 71 and the Department of Transportation's (DOT) 49 CFR Parts 172-177, the shipment of any radioactive materials must meet their requirements [130,132]. The NRC regulations approve the packaging of radioactive materials and require that DOT safety regulations for the transportation of hazardous materials are followed. The safety of all radioactive

material shipments currently occurring around the country is mainly ensured through the use of appropriate shipping containers [132]. Three types of containers are used, industrial, Type A, and Type B, and the use of each type depends on the activity, type, and form of the material [130,132]. Each container type should adequately shield against the material's radiation levels, retain its contents, and maintain sub-criticality within the material configuration [133]. As for radiation levels, they should stay below 2 mSv/h at any point on the surface of the container, or up to 10 mSv/h while meeting additional criteria [132]. If a radioactive material does not meet these packaging requirements and other NRC and DOT regulations, it cannot be shipped. The issue for microreactors is that there currently are no approved containers for shipping entire reactor systems, especially if they have fuel enrichments between 5-20%, as most microreactor designs do [130]. These microreactor designs contain only U^{238} and U^{235} at HALEU levels as their radioactive materials. Although information on the uranium weight loadings vary between design and are is not readily available, U^{238} has a specific activity of 3.36×10^{-7} Ci/g and U^{235} has a specific activity of 2.16×10^{-6} Ci/g [134]. Since typical Light Water Reactor fresh fuel bundles use Type A containers, a modified Type A container that ensures radiation safety and criticality safety is most likely best suited for transporting microreactors [130]. However, additional layers of safety can also be added during transportation by introducing sensors. Although the future microreactor containers will have to be tested and certified that they meet all current regulations, radiation detectors outside the package but within the transport vehicle can monitor radiation levels to confirm the container's integrity during transportation. If the detectors sense higher counts than the accepted levels, it could

indicate to shielding failures within the package created during transportation. Identifying shielding failures when they happen allows for an incidence response to be implemented as soon as possible to reduce the exposure of radiation to the shipping crew and public. It is important to note that this detector example only aims at highlighting the capabilities of deep learning techniques coupled with sensor systems and does not aim at developing said sensor system.

Along with an adequate sensor system, deep learning data analysis techniques can be used to effectively detect any anomalies in the radiation exposure within the microreactor shipping vehicle. Unlike the supervised learning techniques implemented for the CubeSat surveillance system, the microreactor transportation example is suited better for unsupervised learning techniques. As was mentioned in Section 5, unsupervised learning methods of deep learning do not require training on labeled data. In unsupervised learning, relationships and representations can be formed directly from the data. Similar to how classification tasks were a subset of supervised learning, anomaly detection is a subset of unsupervised learning. When monitoring radiation levels, anomaly detection techniques facilitate the identification of any unexpected changes.

To illustrate the application of the technology in this example, a convolutional neural network is used to identify any changes in radiation levels outside the microreactor container. To simulate the radiation detection, it is assumed that the detectors receive a signal described by Eq. 7-1. The detectors receive 100 signals (y) per second. Even though unsupervised learning does not require data labels, it still needs a large dataset to train on. Therefore, the convolutional autoencoder used for anomaly detection is trained on two

hours of nominal signals. For the sake of the example, the sensor system would be used at the fabrication plant to collect two hours' worth of radiation readings before shipping the microreactor in order to train the deep learning algorithm.

$$y = \cos(\pi\xi_1) + \xi_2 \tag{7-1}$$

Where,

$y =$ The signal received by the detectors
$\xi_1 =$ A random number between 0 and 1
$\xi_2 =$ A random number between 0 and 1

As mentioned, the deep learning structure used to identify the anomalous signals is referred to as a convolutional autoencoder. This neural network uses convolutional layers to reduce the input data into a latent space. The latent space is a reduced order representation of the input data. In the CNNs developed in Section 6, the networks reduced an input tensor of size (100, 100, 1) to an output tensor of size (*N*) which represented the predictions for classifying the data. That output tensor of size (N) is also referred to as the latent space. Instead of outputting classification probabilities, however, the latent space of an autoencoder for anomaly detection is used to identify any outliers in the data. When training on the data, the autoencoder is set up as a convolutional U-Net. The network is referred to as a "U-Net" because it encodes the data into a latent space and then decodes the latent space into the original size. The decoder section of the network learns to recreate the original input data from only what is left in the latent space of the encoder. However, when testing the network on new data, only the encoder section of the network is used. Figure 75 illustrates the architecture of the convolutional U-Net. It is important to note that the convolutional and max pooling layers are only one dimension since the input

tensor is only one dimension as well. The input shape of the training data is (7200, 100, 1) representing the two hours of collecting the 100 signals per second. As for a validation dataset, 10% of the input training data is partitioned from validation. Also, the Adam optimizer and Mean Squared Error loss function were used to compile the network architecture. The model was then trained for 100 epochs with a batch size of 32.

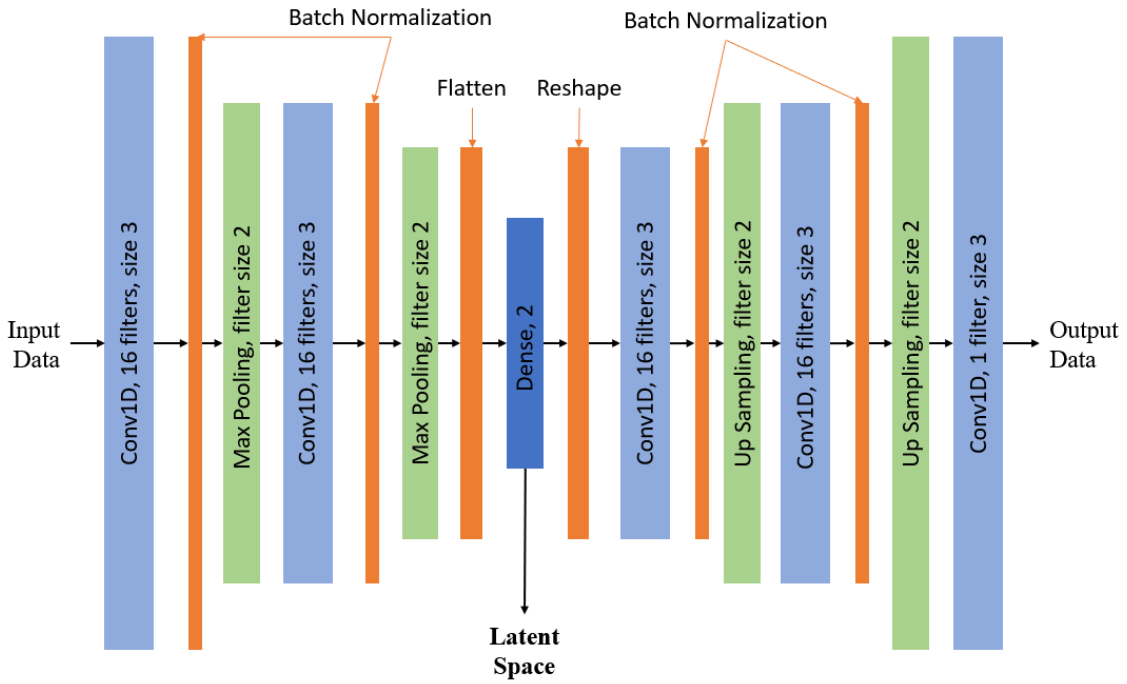


Figure 75. The autoencoder U-Net architecture.

Once the convolutional autoencoder was trained, it was tested on an hour and five minutes of signals. As it translates to the example, the detectors collected data during the first hour and five minutes of transportation. During the last five minutes of detection, the microreactor shielding failed due to the movement of the transport vehicle and the detector

system started receiving higher levels of radiation. Figure 76 shows the magnitude of the signals received by the detectors. The hour and five minutes of signals are then put through the autoencoder. Figure 77 shows the distribution of the latent space created by the network. The graph plots the two numbers outputted by the Dense layer seen in Figure 75. As can be seen, there are two separate clusters of signals within the latent space, indicating anomalous signals. Figure 78 plots the signals corresponding to the points in the smaller latent space cluster over the original signals received. In other words, the seconds at which the signal was determined anomalous by the convolutional autoencoder are specified in Figure 78. As can be seen, the neural network accurately identifies a change in the signals received by the detectors in the microreactor shipping containment. Since a change in radiation levels was detected by the deep learning algorithm, the proper safety measures can be taken to minimize exposure to the transport crew or the public.

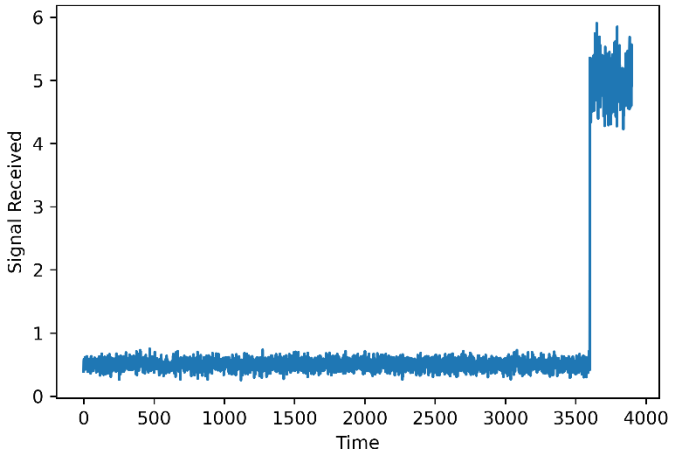


Figure 76. The total signals received by the detectors.

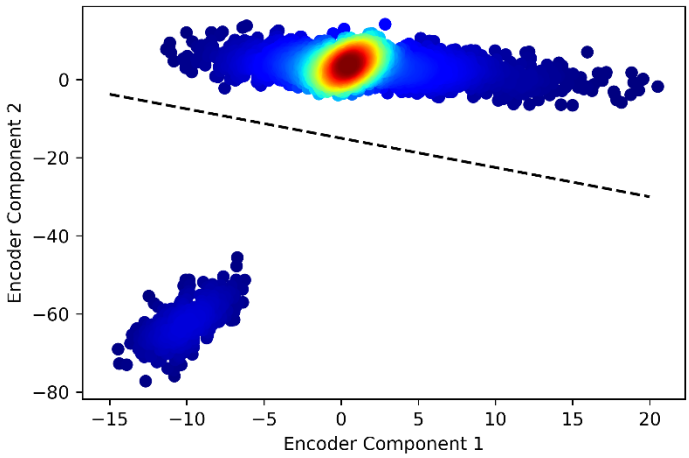


Figure 77. A visual representation of the latent space of the network.

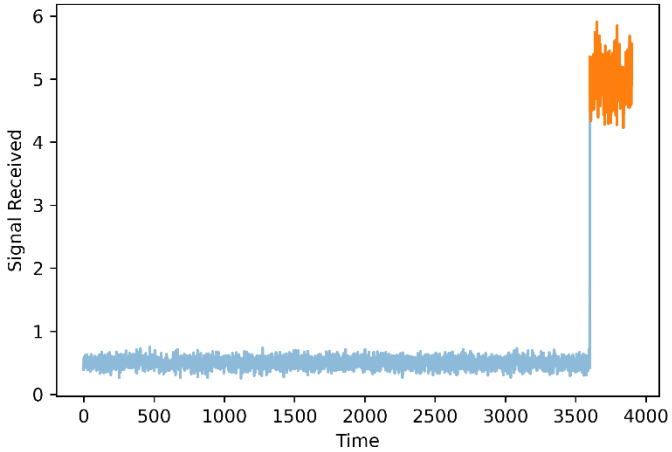


Figure 78. The anomaly cluster points plotted over the complete detector signal.

7.4. Microreactor Applications Conclusion

Even though microreactors present an attractive power generation option for multiple scenarios, they still need to overcome large obstacles on their path toward deployment. Numerous safety and regulatory concerns arise from the desired

transportability and autonomy of these reactors. As was seen, the CubeSat surveillance method developed in this thesis can be adapted directly or indirectly to help overcome such challenges. Directly, the constellation of CubeSats could monitor the integrity and presence of phenomena of interest during the transportation of microreactors or once installed at a plant. Indirectly, some aspects of the CubeSat methodology have the potential of adaptability to other applications for microreactor monitoring, namely its deep learning techniques. An example was explored where anomaly detection techniques of deep learning were used to identify a change in radiation levels during the transportation of a microreactor.

8. CONCLUSIONS

8.1. Thesis Objectives

Cube satellites (CubeSats) present a unique new platform for monitoring localized processes anywhere within the Earth's surface or atmosphere using a novel data analysis technique. Areas of interest can be targeted at certain times on an on-demand basis by storing the CubeSat constellation onboard the International Space Station in what is referred to as "stash and deploy". CubeSats equipped with adequate sensors and data analytics capabilities create an autonomous characterization surveillance method for phenomena of interest. CubeSats are advantageous over conventional satellites for remote monitoring because of their reduced costs and higher simplicity due to the availability of commercially-of-the-shelf components. The work presented in this thesis established a foundation for a CubeSat surveillance system methodology. The following three previously mentioned objectives for the thesis were accomplished throughout the seven previous sections of the thesis:

4. Development and compilation of representative surrogate data sets.
5. Conceptual development of a methodology for heterogeneous data analytics.
6. Illustrative applications of the methodology.

The first objective of this thesis focused on the data analytics structure of the CubeSat system for the recognition of the phenomena of interest. Since the characterization methodology developed was based on machine learning techniques, an adequate dataset was needed to successfully train the model to predict the values of new datasets. The surrogate dataset created in Section 5 also aimed at replicating the diversity of data collected by a multi-sensor system. The second objective established the characterization

method which the CubeSat system will used for analyzing phenomena based on the data set constructed by the previous objective. Using the diverse types of data as input, the deep learning data analytics capabilities made predictions on the kinds of phenomena observed by the CubeSat sensors. The final objective of the thesis was an extension of the first two objectives. It illustrated how the data analytics model developed was implemented in a realistic scenario while incorporating all other aspects of a CubeSat surveillance system. Before the completion of the objectives, extensive considerations were made about the orbital mechanics, satellite architectures and scenarios, and sensor selection for the system.

The three thesis objectives were met mainly by Sections 5 and 6. As was seen, Section 5 created simplified but representative datasets of images from all the different sensor types onboard the CubeSat surveillance system. Each dataset contained images with the phenomena corresponding to each specific sensor type. The presence, size, location, and orientation of phenomena in each image was highly randomized to ensure the characterization methodology only learned features specific to each phenomenon. As was seen, Section 6 first completed the second thesis objective by developing a characterization methodology implemented on all datasets. The characterization methodology included a convolutional neural network for each dataset that outputted a probability of presence for the phenomena in each dataset image. The probabilities are then combined to produce a final characterization solution. The last objective was met at the end of Section 6 when the Tehran example was discussed. Each feature from the CubeSat surveillance system discussed in the thesis was implemented to observe

suspicious activity at an enrichment facility near Tehran and characterize the scene from Low Earth Orbit. The following subsection discusses in detail the conclusions produced from each section of the thesis.

8.2. Thesis Overview

As mentioned, the topics covered in the previous seven sections of the thesis contributed to the completion of the three thesis objectives. Sections 1-6 each focused on a different aspect of the CubeSat surveillance system design process methodology, as seen in Figure 79, with Section 6 covering both the creation of the data analytics methodology and illustrative application. Explicitly, Section 5 directly completed the first of the three thesis objectives, while Section 6 fulfilled the last two.

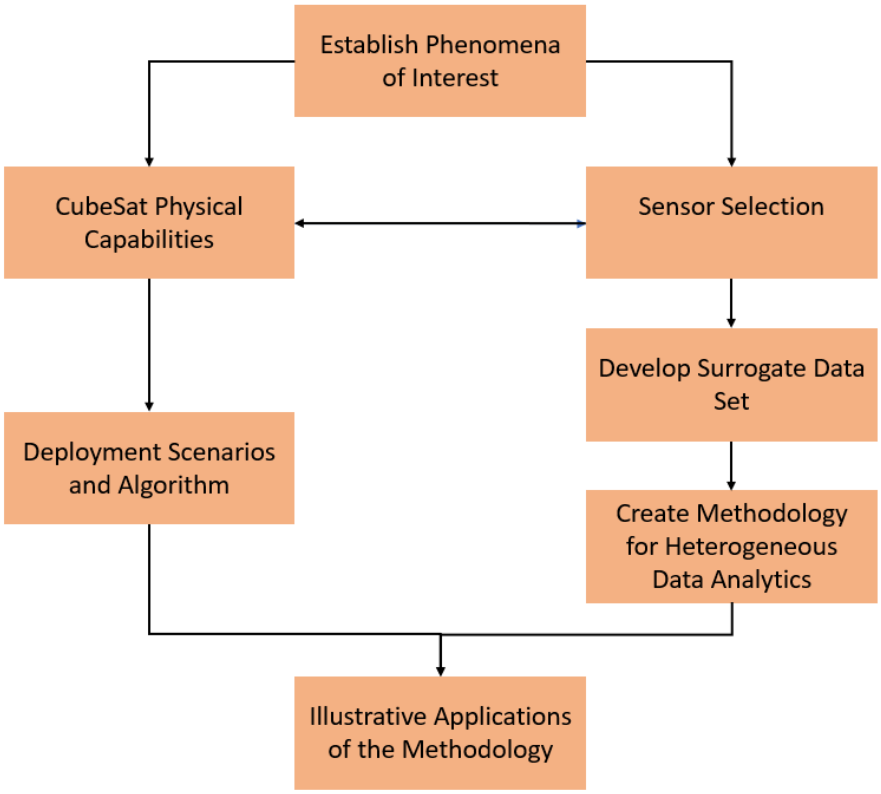


Figure 79. A free body diagram of the design process used to develop the methodology for the CubeSat system.

The thesis opened with Section 1 (Introduction). The section first introduced the CubeSat surveillance system and established the need for such a system. Due to lower costs and higher simplicity, constellations of CubeSats present an attractive remote monitoring system as applied to the use example of nuclear fuel cycle surveillance. The phenomena of interest for the surveillance system were introduced. The first step in developing a surveillance system is identifying what type of phenomena are of interest for observation. Defining the phenomena and their characteristics will determine everything else about the surveillance system, like the type of sensors needed and the physical CubeSat architecture. Due to the nature of miniature satellites in orbit, it was determined

that the CubeSat system was best suited for events of immediate interest. Three different categories of phenomena of interest surrounding the nuclear fuel cycle were identified for the nuclear surveillance use example. The three categories were: Vehicles of Interest, Facilities and Infrastructural Emergencies of Interest, and Construction and Mining Events of Interest. For each category, different parameters and signatures were specified for the phenomena. Identifying the signatures aids in determining the sensor requirements for the surveillance system. Then, Section 1 closed by stating the objectives for the thesis and gave an overview of each section to follow.

The goal of Section 2 (CubeSat-based Surveillance Platform Capability Assessment) was to explore the physical feasibility of a CubeSat platform for a surveillance system. First, the section provided a more specific definition of a cube satellite and explored its capacities for supporting a surveillance platform. The major components and capabilities of a CubeSat and the nature of an ISS deployment were defined. Next, a summary of a general CubeSat design process was explored for facilitating future. The design processes outlined in CubeSat designed guides were analyzed and adapted to the CubeSat surveillance system. Other CubeSat mission design procedures were also mentioned as reference points. The individual types of components needed for the CubeSat surveillance system mission were identified along with their COTS feasibility. The CubeSats' payloads, or sensors, were determined as the only non-COTS components. Figure 80 shows a schematic of the different components required for an individual 3U CubeSat. Also, it was determined that the CubeSat system would feature a combination of 3U and 6U sized satellites depending on the size of the payload. To close

out the section, a CubeSat's orbital capabilities in low Earth orbit (LEO) were examined through modelling. The General Mission Analysis Tool from NASA was used to observe the trajectory of a CubeSat's flight and analyze the kind of access it can have to targets on the surface.

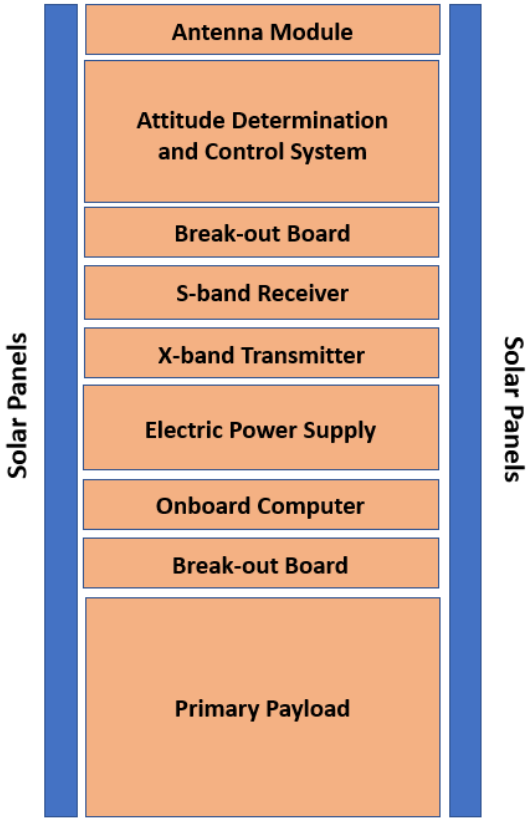


Figure 80. A schematic of the components for a 3U CubeSat in the CubeSat surveillance system.

Section 3 (Architecture and Deployment Scenarios) explored different architecture and deployment scenarios for the CubeSat system. The system was determined to be comprised of a constellation of multiple satellites instead of one. Table 59 illustrates the advantages and disadvantages of a single or multiple satellite surveillance system.

Although a constellation architecture has its drawbacks, the main reasoning for the decision was driven by a constellation's higher characterization accuracy due to a higher number of sensors in the system. Since a constellation was chosen for the system, different communications configurations were explored. Table 60 lists the decision matrix used when selecting the ideal communications option for the system. Intersatellite communication using a satellite relay network, like Globalstar, combined with a network of ground stations, like KSAT was determined as the optimal communications option. Also, a satellite system architecture featuring a single CubeSat as the major computational unit for processing most of the data collected by the sensors was chosen. The last topic of the section introduced a mathematical algorithm which utilizes GMAT for determining CubeSat deployment times.

Table 59. The advantages and disadvantages between a 1 CubeSat or constellation system.

Attribute	1 CubeSat	Constellation of CubeSats
Higher Characterization Accuracy Through Sensor Diversity		✓
Lower Cost	✓	
Increased System Security		✓
Robustness Through Sensor Redundancy		✓
Simplicity	✓	
Longer Overall Access Times to Ground		✓

Table 60. The decision matrix for the CubeSat constellation communications options.

	Ground Station Only	Inter-CubeSat	Additional Satellite Relay
Operational Complexity	Moderate	High	High
Data Flow Steps	High	Moderate	Moderate
Speed to Surface	Moderate	High	High
Autonomy	Low	High	High
Lifetime	High	Low	High
Cost	Moderate	Moderate	High

Section 4 (Sensor Selection) first explored the capabilities of remote monitoring and considered the surveillance requirements for the phenomena of interest. Defining the signatures for the phenomena determines the type of sensors needed for the CubeSat surveillance system. For each phenomenon of interest, their signature resolutions in the spectral, spatial, and temporal dimensions were established. Table 61 shows the surveillance requirements for the different phenomena. It is important to note that the terms for spectral and temporal resolution used in the table are lose interpretations of their actual meaning when it comes to sensor resolutions. Then, the different sensor requirements necessary for monitoring the signatures were explored, and the various kinds of sensors and their capabilities were defined. It was determined that the CubeSat surveillance system can accomplish its observation goals by featuring a combination of panchromatic and multispectral sensors operating in the visible and near-infrared spectrum, multispectral sensors operating in the infrared spectrum, hyperspectral sensors operating in the infrared spectrum, and multispectral sensors operating in the ultraviolet spectrum. The current capabilities and CubeSat heritage for the recommended sensors

were explored, and it was observed that future sensor development is needed to reach the necessary resolution and size requirements, as seen in Table 62. The section concludes with an analysis of CubeSat satellite architecture capabilities for housing sensors regarding size and power demands.

Table 61. The sensor type recommendations for observing each parameter.

Phenomenon	Parameter	Sensor Recommendation
Automobiles and Airplanes	Length, Width, Height	Panchromatic and Multispectral
	Speed	Panchromatic and Multispectral
	Temperatures	Multispectral
	Gas Emissions	Hyperspectral
Facilities and Emergencies	Length, Width, Height	Panchromatic and Multispectral
	Temperatures	Multispectral
	Gas Emissions	Hyperspectral
	Aerosol Index	Multispectral
Construction and Mining	Length, Width, Height	Panchromatic and Multispectral
	Speed	Panchromatic and Multispectral
	Temperatures	Multispectral
	Gas Emissions	Hyperspectral
	Footprint	Panchromatic and Multispectral

Table 62. Sensor type recommendations and CubeSat heritage at required spatial resolutions.

Phenomena	Parameter	Sensor Type	CubeSat Heritage	CubeSat Heritage at Spatial Resolution	Future Sensor Development Needed?
Automobiles and Airplanes	Length, Width, Height	Panchromatic/ Multispectral	Yes	Yes	No
	Speed	Panchromatic/ Multispectral	Yes	Yes	No
	Temperatures	Multispectral	Yes	No	Yes
	Gas Emissions	Hyperspectral	Yes	No	Yes
Facilities and Infrastructural Emergencies	Length, Width, Height	Panchromatic/ Multispectral	Yes	Yes	No
	Temperatures	Multispectral	Yes	No	Yes
	Gas Emissions	Hyperspectral	Yes	No	Yes
	Aerosol Index	Multispectral	Yes	No	Yes
Construction and Mining Events	Length, Width, Height	Panchromatic/ Multispectral	Yes	Yes	No
	Speed	Panchromatic/ Multispectral	Yes	Yes	No
	Temperatures	Multispectral	Yes	No	Yes
	Gas Emissions	Hyperspectral	Yes	No	Yes
	Footprint	Panchromatic/ Multispectral	Yes	Yes	No

After the selection of sensors was accomplished Section 5 (Surrogate Dataset) directly accomplished the first objective of the thesis: Development and compilation of representative surrogate datasets. The dataset created in this section for training the deep learning model for characterization aimed at representing the data collected from the

different sensors on board the surveillance system at a proof of concept level. Rather than training the methodology on real satellite imagery, the dataset is a simplified representation. As can be seen in Table 62, since four different sensor types were chosen, four different datasets were created. The different datasets are referred to as the Dimension, Temperature, Gas, and Aerosol datasets. Each dataset was created using the same methodology of producing 100 x 100 pixel "images" which include the phenomena of interest. The only difference between datasets was the *base* value added to each pixel and each dataset's corresponding phenomena list. The dataset creation methodology used a multitude of random numbers for determining the inclusion of phenomena in each "image". Including randomness in the creation of the "images" was essential for ensuring the characterization algorithm only learned features directly indicating the presence of phenomena. By including phenomena in the data points in random locations and in random amounts allows the deep learning algorithm to identify each phenomenon in any condition in future data. Each "image" created by the dataset included labels indicating the presence or absence of a phenomenon. The length of the data labels varied per dataset. The use of labels for the datasets are what allow the characterization algorithm to develop relationships in the data to accurately predict the labels of future data. Figure 81 shows an example image from the Dimension dataset and its corresponding label. Also, Section 5 introduced the general concepts and math behind deep learning and convolutional neural networks.

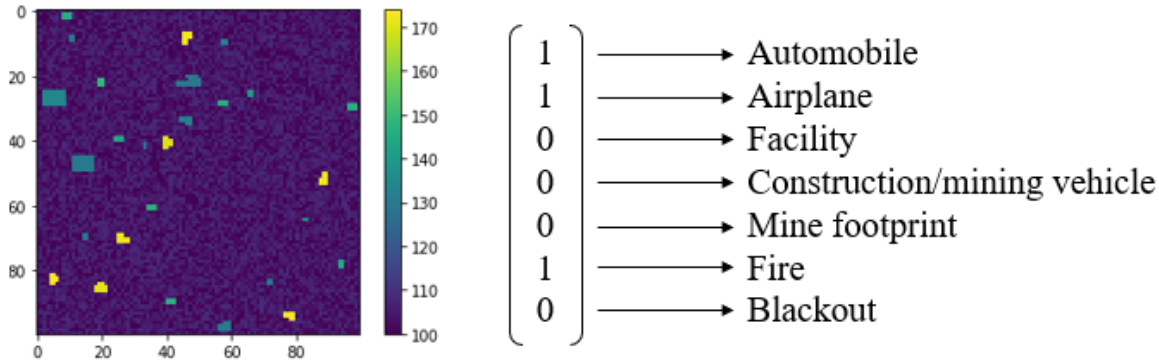


Figure 81. An example image created in the Dimensions dataset.

Section 6 (Characterization Methodology) used the surrogate dataset created in the previous section to address the final two objectives of the thesis: Conceptual development of a methodology for heterogeneous data analytics and Illustrative application of the methodology. The section introduced the characterization methodology developed for the identification of phenomena of interest from the CubeSat surveillance platform, as seen in Figure 82. The characterization methodology consisted of four convolutional neural networks trained for each sensor type onboard the CubeSat system. The CNNs were trained on the four surrogate datasets representing simplified versions of satellite imagery. Different numbers of layers, filters, nodes, and filter sizes were explored to optimize the CNN architecture. During training, several different hyperparameter values were explored to produce the best performing model for each dataset. While test accuracy on test data subset was the performance metric used to compare different CNN architectures during optimization, it does not describe well the true performance of a CNN at predicting the presence of specific phenomenon in an "image". Other metrics, like Recall Rate and Precision Rate, were introduced and calculated. Table 63 restates the performance of the

CNNs on the phenomena for each dataset. Within the overall characterization methodology, the predictions from each CNN are combined in the last step to produce a final characterization solution to be transmitted by the CubeSat system. The section wrapped up by discussing an illustrative application in which the CubeSat surveillance system was tasked with investigating suspicious activity at an enrichment facility near Tehran. Figure 83 illustrates the overall data flow through each aspect of the CubeSat system. The topics from Section 1 were discussed to determine the type of phenomena the CubeSat system was looking for in the example. Topics from Section 2 and 3 were discussed to illustrate how the CubeSat system architecture and orbital mechanics are applied to the example. Access times for the CubeSat to Tehran were calculated using the algorithm from Section 3 and GMAT. It was discussed how the sensor types selected in Section 4 would produce satellite images of the target area, which were represented by images created using the methodology in Section 5. Then, the characterization methodology was used to produce final predictions of phenomena from the "images" taken by the sensors. Table 63 compares the true phenomena probabilities to the predicted phenomena probabilities. The predicted phenomena probabilities characterizing the observed location are then sent to the surface.

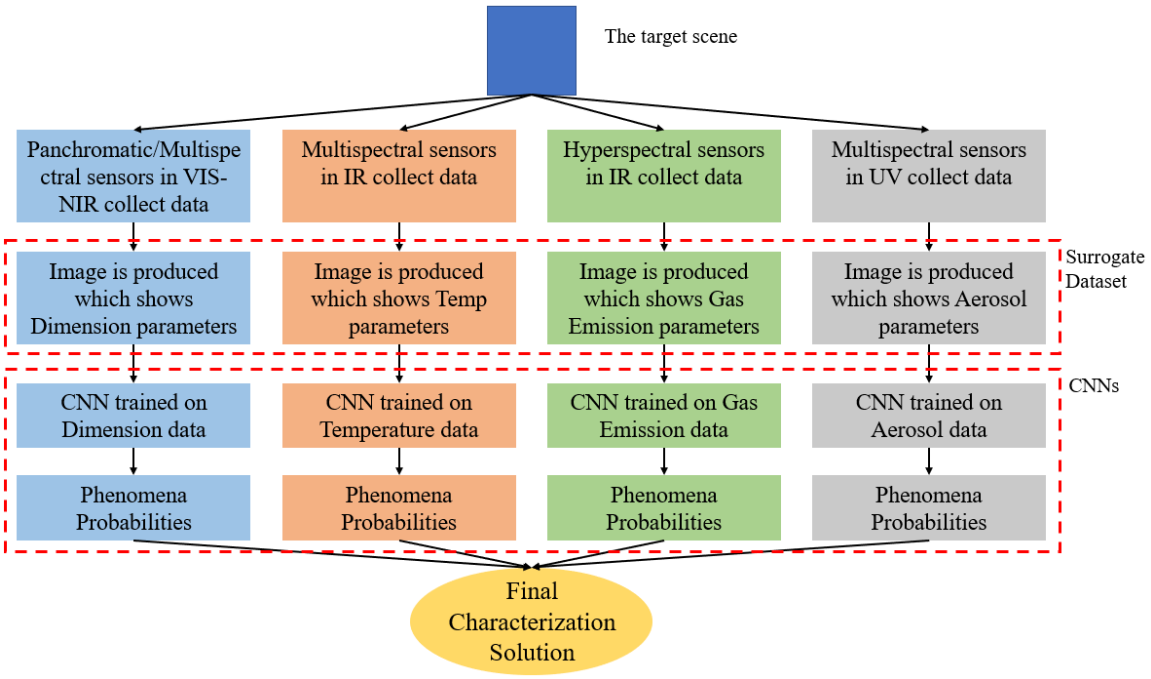


Figure 82. Structure of data flow in characterization methodology.

Table 63. The performance of each CNN on predicting phenomena and the final characterization solution probabilities for the Tehran application.

The performance of the Dimension CNN for identifying phenomena of interest.

Phenomena	Recall Rate	Precision Rate	FP Rate	TN Rate	FN Rate
Automobile	92.5%	99.9%	0.08%	99.9%	7.47%
Airplane	96.1%	99.2%	0.77%	99.2%	3.93%
Facility	98.0%	98.5%	1.43%	98.6%	1.97%
Cons/Mine Vehicle	96.0%	99.8%	0.16%	99.8%	4.01%
Footprint	99.6%	99.9%	0.08%	99.9%	0.44%
Fire	93.7%	97.4%	2.54%	97.5%	6.34%
Blackout	95.4%	98.0%	2.03%	98.0%	4.58%

The performance of the Temperature CNN for identifying phenomena of interest.

Phenomena	Recall Rate	Precision Rate	FP Rate	TN Rate	FN Rate
Automobile	90.0%	99.8%	0.16%	99.8%	10.0%
Airplane	94.1%	99.5%	0.43%	99.6%	5.86%
Fire	93.9%	96.8%	2.85%	97.1%	6.06%
Blackout	90.6%	92.3%	7.95%	92.1%	9.41%
Cons/Mine Vehicle	97.2%	99.3%	0.73%	99.3%	2.77%
Cons/Mine Process	89.7%	99.8%	0.20%	99.8%	10.3%

The performance of the Gas CNN for identifying phenomena of interest.

Phenomena	Recall Rate	Precision Rate	FP Rate	TN Rate	FN Rate
Automobile	95.6%	99.7%	0.33%	99.7%	4.35%
Airplane	97.4%	98.8%	1.23%	98.8%	2.58%
Fire	95.0%	99.9%	0.12%	99.9%	4.95%
Cons/Mine Vehicle	97.1%	99.5%	0.52%	99.5%	2.92%
Cons/Mine Process	99.1%	99.6%	0.41%	99.6%	0.91%

The performance of the Aerosol CNN for identifying phenomena of interest.

Phenomena	Recall Rate	Precision Rate	FP Rate	TN Rate	FN Rate
Fire	96.9%	99.8%	0.16%	99.8%	3.08%
Cons/Mine Process	99.3%	99.9%	0.12%	99.9%	0.74%

The final characterization solution for the Tehran enrichment facility.

Probability	**Automobile**	**Airplane**	**Facility**	**C/M V**
True	1	0	1	1
Predicted	0.910	0.005	1	1
	C/M P	**Footprint**	**Fire**	**Blackout**
True	1	0	0	0
Predicted	0.999	0	0.007	0.042

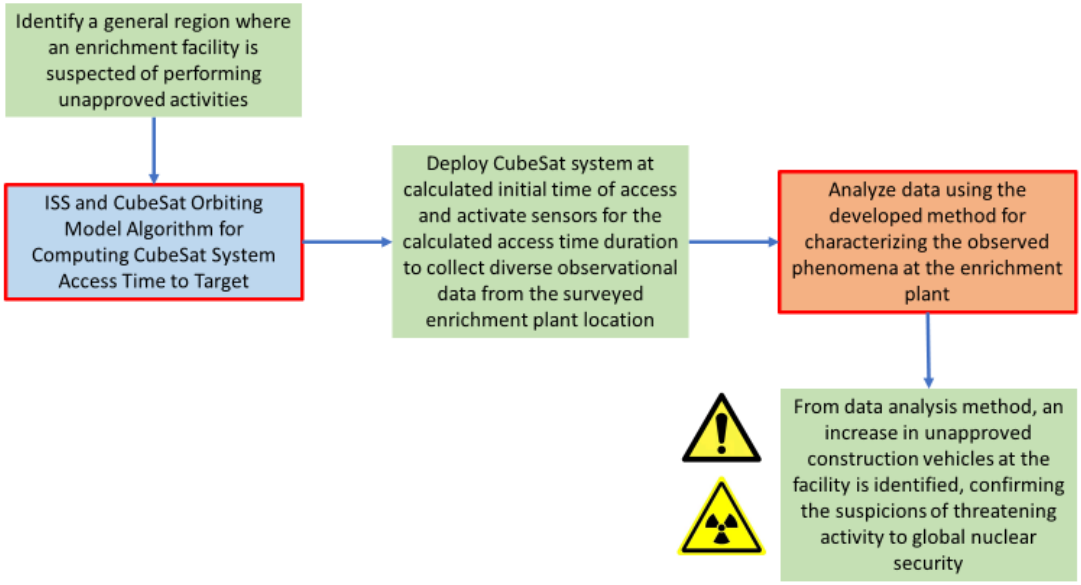


Figure 83. The data flow for observing and identifying suspicious activity at an enrichment plant.

The last section, Section 7 (Microreactor Applications), explores the applicability of the CubeSat surveillance system towards aiding microreactor deployment. Even though microreactors present an attractive power generation option for multiple scenarios, they still need to overcome large obstacles on their path toward deployment. Numerous safety and regulatory concerns arise from the desired transportability and autonomy of these reactors. As was seen, the CubeSat surveillance method developed in this thesis can be adapted directly or indirectly to help overcome such challenges. Directly, the constellation of CubeSats could monitor the integrity and presence of phenomena of interest during the transportation of microreactors or once installed at a plant. Indirectly, some aspects of the CubeSat methodology have the potential of adaptability to other applications for microreactor monitoring, namely its deep learning techniques. An example was explored

where anomaly detection techniques of deep learning were used to identify a change in radiation levels during the transportation of a microreactor.

8.3. Future Work

As was mentioned multiple times throughout the thesis, a final CubeSat surveillance system was not developed, only the basis for its methodology. Therefore, there is a lot of future work than can be done to expand on the work presented in this thesis. There are four potential avenues in which the work can be immediately expanded and turned into future projects. Those options are:

1. CubeSat construction.
2. Sensor development.
3. Create characterization methodologies on real satellite data.
4. Expand on microreactor applications.

The first step in the future development a CubeSat surveillance system is to construct the actual satellites for the constellation. Section 2 formed the foundation to be followed for physical construction of the CubeSats. All the components listed in the section can be directly bought from vendors and assembled into a final product. As was also seen in Section 2, the CubeSat design process may take between 1-2 years. Getting through the licensing process, integrating the different components into the CubeSat bus, material and environmental testing, and communications networks setup all takes time and would constitute as its own individual project.

Even if a full constellation of CubeSats is constructed, they will not successfully observe the phenomena of interest without adequate sensors. As was seen in Section 4, only one out of the four different sensor types recommended for the surveillance system have an immediate technology readiness level. Multispectral sensors in the infrared

region, hyperspectral sensors in the infrared region, and multispectral sensors in the ultraviolet region need to be developed to image at the required spatial resolutions for the phenomena of interest while still fitting inside a CubeSat form factor. A whole industry is built around furthering sensor technology, and the development of each sensor mentioned would constitute a whole project alone.

As was mentioned in the Limitations subsection of Section 6, the characterization methodology developed in this thesis cannot characterize actual satellite images. The solution to developing an accurate characterization methodology for true CubeSat images is to launch an exploratory CubeSat mission with identical satellite architectures and sensors with the goal of collecting as many images of phenomena of interest as possible. The first CubeSat system mission would only serve the purpose of data collection. The collected images can then be used to train the characterization methodology for a second CubeSat mission. Therefore, creating a truly accurate characterization methodology for a CubeSat surveillance system can only be accomplished after the first to avenues for future work have been explored.

The final avenue for future work of this thesis was introduced in Section 7, with the focus mainly on indirect application of the CubeSat methodology. Some aspects of the CubeSat methodology have the potential of adaptability to other applications for microreactor monitoring, namely its deep learning techniques. Machine/deep learning algorithms coupled with sensor systems within and around a microreactor can be used to improve the safety of its transportation and autonomous operation. Although the CubeSat characterization methodology was limited to supervised learning techniques, a

microreactor monitoring system does not possess the same limitations. Unsupervised learning techniques for machine learning have shown effectiveness in anomaly detection situations. Creating a microreactor monitoring system with adequate sensors and data analytics capabilities presents a very interesting topic for future projects.

REFERENCES

[1] “Achieving Science with CubeSats: Thinking Inside the Box”. May 2016, Space Studies Board, Div. Eng. Phys. Sciences, Nat. Academics of Science, Engineering and Medicine, 2016.

[2] C. NIETO-PEROY and M. R. EMAMI, “CubeSat Mission: From Design to Operation”, *Applied Sciences, 9* (2019) 3110. doi: 10.3390/app9153110

[3] H. B. MARTIN et al, “Bolstering Mission Success: Lessons Learned for Small Satellite Developers Adhering to Manned Spaceflight Requirements”, NanoRacks LLC, *Proc. of 32nd Annual AIAA/USU Conference on Small Satellites* (2018).

[4] B. JASANI et al., *International Safeguards and Satellite Imagery,* Springer-Verlag Berlin Heidelberg, pp. 1-8, (2009).

[5] US Department of Energy National Nuclear Security Administration, “Prevent, Counter, and Respond – NNSA’s Plan to Reduce Global Nuclear Threats FY 2020-FY 2024”, *Report to Congress,* (2019).

[6] R. QUEVENCO, “Completing the Picture: Using Satellite Imagery to Enhance IAEA Safeguards Capabilities”, *IAEA Bulletin,* pp. 24-25, (2016).

[7] K. CHITUMBO, J. BUNNEY, G. LEVE, and S. ROBB, “Satellite Imagery and the Department of Safeguards”, International Atomic Energy Agency, IAEA-SM—367, (2001).

[8] J. HSU, “How Experts Comb Satellite Images for Clues on North Korea’s Nuclear Tests”, IEEE Spectrum, 29 Sep 2017; https://spectrum.ieee.org/tech-talk/aerospace/satellites/how-experts-comb-satellite-images-for-clues-on-north-koreas-nuclear-tests (current as of Nov. 17, 2020).

[9] T. S. KELSO, “CelesTrak”, CelesTrak; https://www.celestrak.com/ (current as of Nov. 17, 2020).

[10] P. ANDERSON et al, “The ISS as a Launch Platform for Phenomena of Interest”, Space Dynamics Laboratories, Utah State University Research Foundation, (2011).

[11] F. R. HOOTS, “Reformulation of the Brouwer Geopotential Theory for Improved Computational Efficiency”, *Celestial Mechanics,* **24**, 367-375, (1981).

[12] R. SCHWARZ, “Keplerian Orbit Elements to Cartesian State Vectors (Memorandum No. 1)”, Memorandum Series, (2017).

[13] “New Car Dimensions in the European Market with the Photo of Each Automobile Size Showing Length, Width and Height”, Automobiledimension.com; https://www.automobiledimension.com (current as of Nov. 17, 2020).

[14] “Federal Size Regulations for Commercial Motor Vehicles”. US Department of Transportation Federal Highway Administration, 2004; https://ops.fhwa.dot.gov/freight/publications/size_regs_final_rpt/ (current as of Nov. 17 2020).

[15] R. WORLEDGE, “Free Flow Vehicle Speed Statistics: Great Britain 2012”. Vehicle Speeds Statistics, UK Department for Transport, (2013).

[16] M. BASTAN et al, “Remote Detection of Idling Cars Using Infrared Imaging and Deep Networks”. Nanyang Technological University, (2018).

[17] “Greenhouse Gas Emissions from a Typical Passenger Vehicle”. US Environmental Protection Agency, 2018; https://www.epa.gov/greenvehicles/greenhouse-gas-emissions-typical-passenger-vehicle (current as of Nov. 17, 2020).

[18] “737 Airplane Characteristics for Airport Planning”. Boeing Commercial Airplanes, D6-58325-6, (2013).

[19] “Backgrounder”, Boeing Commercial Airplanes Communication, (2014); http://www.boeing.com/farnborough2014/pdf/BCA/Bck%20-%20NG%20737%20 Family_June%202014.pdf (current as of Nov. 17, 2020).

[20] S. ACKERT, “Engine Maintenance Management”, presented at Managing Technical Aspects of Leased Assets, Madrid, Spain, 12 May 2015. Jackson Square Aviation.

[21] “A380 Aircraft Characteristics Airport and Maintenance Planning AC”, Airbus S.A.S, (2005).

[22] “Emirates A380 Specifications”, The Emirates Group; https://www.emirates.com/ english/flying/our_fleet/emirates_a380/emirates_a380_specifications.aspx. (current as of Nov. 17, 2020).

[23] “Type-Certificate Data Sheet, EASA.E.012 for RB211 Trent 900 series engines”, Issue 10, European Union Aviation Safety Agency, 22 Nov. 2019.

[24] “Technical Order 00-105E-9 and STANAG 3896”. Aerospace Emergency Rescue and Mishap Response Information, 1 Feb. 2006. Segment 11.

[25] S. AFTERGOOD, “US Military Aircraft, Federation of American Scientists”, 2017; https://fas.org/man/dod-101/sys/ac/index.html (current as of Nov. 17, 2020).

[26] “C-5M Super Galaxy”. US Air Force, 15 May 2006; https://www.af.mil/About-Us/Fact-Sheets/Display/Article/104492/c-5-abc-galaxy-c-5m-super-galaxy/ (current as of Nov. 17, 2020).

[27] A. D. ARIYANAYAGAM and M. MAHENDRAN, “Fire Safety of Buildings Based on Realistic Fire Time-Temperature Curves”, *Proceedings of the 19th CIB World Building Congress, Brisbane 2013: Construction and Society*, Brisbane, 2013. Brisbane, Australia: Queensland University of Technology.

[28] “Fires and Thermal Environments”, Thermal Protective Clothing for Firefighters, Elsevier, pp. 5-15, (2017).

[29] “How hot is a Hawaiian volcano?”, US Geological Survey; https://www.usgs.gov/faqs/how-hot-a-hawaiian-volcano?qt-news_science_products=0#qt-news_science_products (current as of Nov. 17, 2020).

[30] T. ROBERTS, G. DAYMA, and C. OPPEMHIEMER, “Reaction Rates Control High-Temperature Chemistry of Volcanic Gases in Air”, *Frontiers in Earth Science* **7** (2019).

[31] BRANZ and SCION, “Assessing the Impact of Vegetation and House Fires on Greenhouse Gas Emissions”, The New Zealand Fire Service Commission, Fire Research Report, (2010).

[32] “OMI aerosol measurements: AI (OMI Aerosol Index)”, Ozone Monitoring Instrument, Koninklijk Nederlands Meteorologisch Instituut, 20 Sep. 2018; https://projects.knmi.nl/omi/research/product/product_generator.php?info=page&product=aerosol&flavour=AI&long=OMI%20Aerosol%20Index (current as of Nov. 17, 2020).

[33] “Ash Plumes”, Earth Observation Data, National Aeronautics and Space Administration, 10 Feb. 2020; https://earthdata.nasa.gov/earth-observation-data/near-real-time/hazards-and-disasters/ash-plumes (current as of Nov. 17, 2020).

[34] C. KUENZER, “Physical Principles of Remote Sensing, Thermal Remote Sensing”, German Remote Sensing Data Center, DFD, German Aerospace Center, DLR. Julius Maimilians Universitat Wurzburg; https://earth.esa.int/documents/973910/1002056/CK3.pdf/4e5b4e5a-d898-43b8-9e5c-ba7494aa58c8 (current as of Nov. 17, 2020).

[35] S. ANUPOJU, “Types of Heavy Construction Equipment”, The Constructor; https://theconstructor.org/construction/heavy-construction-equipment-types/26305/ (current as of Nov. 17, 2020).

[36] “Equipment Specs and Charts”, ConstructionEquipmentGuide.com; https://www.constructionequipmentguide.com/equipment-specs-and-charts (current as of Nov. 17, 2020).

[37] “800”, CAT, Caterpillar; https://www.cat.com/en_US/products/new/equipment/draglines/draglines/18265906.html (current as of Nov. 17, 2020).

[38] “Excavators 320 GC”, CAT, Caterpillar; https://www.cat.com/en_US/products/new/equipment/excavators/medium-excavators/1000032601.html (current as of Nov. 17, 2020).

[39] “Wheel Tractor-Scrapers 637K”, CAT, Caterpillar; https://www.cat.com/en_US/products/new/equipment/wheel-tractor-scrapers/coal-bowl-scrapers/1000024172.html (current as of Nov. 17, 2020).

[40] “Wheel Trenchers”, Wolfe Equipment; https://www.wolfeequipment.com/trenchers/ (current as of Nov. 17, 2020).

[41] “Wheel Loaders 950M”, CAT, Caterpillar; https://www.cat.com/en_US/products/new/equipment/wheel-loaders/medium-wheel-loaders/1000029161.html (current as of Nov. 17, 2020).

[42] “Power Tower Cranes MDT 269 J10”, Manitowoc; https://www.manitowoccranes.com/en/cranes/potain/top-slewing/mdt-ccs-tower-cranes/MDT-269-J10 (current as of Nov. 17, 2020).

[43] “Compactors 816K”, CAT, Caterpillar; https://www.cat.com/en_US/products/new/equipment/compactors/landfill-compactors/1000020360.html (current as of Nov. 17, 2020).

[44] “1055 JLG Telehandler”, JLG Industries; https://www.jlg.com/en/equipment/telehandlers/jlg/1055?Cookie=language,language,language (current as of Nov. 17, 2020).

[45] “643L-II Wheeled Feller Buncher”, John Deere, Deere & Company; https://www.deere.com/en/wheeled-feller-bunchers/643l-ii/ (current as of Nov. 17, 2020).

[46] “Off-Highway Trucks 785G (Tier 4 Final/Stage V)”, CAT, Caterpillar; https://www.cat.com/en_US/products/new/equipment/off-highway-trucks/mining-trucks/1000033348.html (current as of Nov. 17, 2020).

[47] “Equipment MD6250”, CAT, Caterpillar; https://www.cat.com/en_US/products/new/equipment/drills/rotary-drills/1000031083.html (current as of Nov. 17, 2020).

[48] “C4.4 ACERT Industrial Engine”, CAT, Caterpillar. LEHH0551-01 (3-13), (2013).

[49] L. M. SNELL, “Monitoring Temperatures in Concrete Construction Using IR Thermometers”. Concrete International, pp. 59-63, Jan. 2015.

[50] B. H. CARY, *Modern Welding Technology,* Upper Saddle River: Prentice Hall, 1998: 466, 108.

[51] W. A. BOWDITCH, *Welding Technology Fundamentals,* Tinley Park: The Goodheart-Willcox Company, Inc. 1997: 269.

[52] J. HONG, G. Q. SHEN, Y. FENG, W. S. LAU, and C. MAO, “Greenhouse Gas Emissions During the Construction Phase of a Building: A Case Study in China”, *Journal of Cleaner Production* **103**, 249-259 (2015).

[53] “’Crystalline Silica Exposure’ Health Hazard Information for General Industry Employees”, US Department of Labor, Occupational Safety and Health Administration, 2002; https://www. osha.gov/Publications/osha3176.html (current as of Nov. 17, 2020).

[54] Uranium Mining in Virginia: Scientific, Technical, Environmental, Human Health and Safety, and Regulatory Aspects of Uranium Mining and Processing in Virginia. Committee on Uranium Mining in Virginia; Committee on Earth Resources; National Research Council. Washington (DC): National Academies Press (US); 2011 Dec 19. 6, Potential Environmental Effects of Uranium Mining, Processing, and Reclamation.

[55] “Uranium Mining in Virginia: Scientific, Technical, Environmental, Human Health and Safety, and Regulatory Aspects of Uranium Mining and Processing in Virginia”, *The National Academies of Sciences Engineering Medicine,* pp. 96-122 (2012).

[56] The Editors of Encyclopedia Britannica, “Nitroglycerin”, Encyclopedia Britannica, Encyclopedia Britannica, Inc. 7 May 2015; https://www.britannica.com/science/nitroglycerin (current as of Nov. 17, 2020).

[57] “What is a Nuclear Microreactor?”, U.S. Department of Energy Office of Nuclear Energy, 23 October 2018; https://www.energy.gov/ne/articles/what-nuclear-microreactor (current as of Nov. 17, 2020).

[58] A. TOORIAN, “CubeSat Design Specification”, version 9, California Polytechnic State University Aerospace Engineering Department, (2004).

[59] “CubeSat 101 Basic Concepts and Processes for First-Time CubeSat Developers”, NASA CubeSat Launch Initiative (Oct. 2017).

[60] “Satellite Technology Powered by The Globalstar Satellite Network”, Globalstar; https://www.globalstar.com/en-us/corporate/about/our-technology (current as of Nov. 17, 2020).

[61] D. GERHARDT and S. PALO, “Passive Magnetic Attitude Control for CubeSat SpaceCraft”, *Proc. of 24th Annual AIAA/USU Conference on Small Satellites,* no. SSC10-VIII-5 (2010).

[62] J. LI, M. POST, T. WRIGHT, and R. LEE, “Design of Attitude Control Systems for CubeSat-Class Nanosatellite”. *Journal of Control Science and Engineering* **2013**, 657182, 1-15 (2013).

[63] “Solar Arrays”, AAC Clyde Space; https://www.aac-clyde.space/satellite-bits/solar-arrays (current as of Nov. 17, 2020).

[64] A. WANN et al., “NanoRacks CubeSat Deployer Clearance Cone Definition”, CAMMP AI: EC-1375 (Aug. 2013).

[65] “3-Unit CubeSat Structure”, Innovative Solutions in Space B.V., 2020; https://www.isispace.nl/product/3-unit-cubesat-structure/ (current as of Nov. 17, 2020).

[66] “Lithium Ion NCR18650B”, Panasonic, Version 13.11 R1, SANYO Energy (USA) Corporation (2012).

[67] “PC/104 Specification”, Version 2.6; PC/104 Embedded Consortium: San Francisco, CA, USA (Oct. 2008).

[68] J. CHAMPAGNE, S. HANSEN, T. NEWSWANDER and B. CROWTHER, “CubeSat Image Resolution Capabilities with Deployable Optics and Current Imaging Technology”, *Proc. of 28th Annual AIAA/USU Conf. Small Satellites*, SSC14-VII-2 (2014).

[69] “Outgassing Search and Report Help”, National Aeronautics and Space Administration, 15 Jan. 1997; https://outgassing.nasa.gov/help/og_help.html (current as of Nov. 17, 2020).

[70] “ISS Trajectory Data”, National Aeronautics and Space Administration, Human Space Flight; https://spaceflight.nasa.gov/realdata/sightings/SSapplications/Post/JavaSSOP/orbit/ISS/SVPOST.html (current as of Nov. 17, 2020).

[71] “Systems Tool Kit (STK)”, Analytical Graphics, Inc, 2020; https://www.agi.com/products/stk (current as of Nov. 17, 2020).

[72] “NASA Software General Mission Analysis Tool (GMAT) v.R2016a”, National Aeronautics and Space Administration, NASA Technology Transfer Program; https://software.nasa.gov/software/GSC-17177-1 (current as of Nov. 17, 2020).

[73] “Buying Optical Satellite Imagery – continued page 2 Off-Nadir Angle/Elevation Angle”, LandInfo Worldwide Mapping LLC, 2018; http://www.landinfo.com/buying-optical-satellite-imagery-2.html (current as of Nov. 17, 2020).

[74] “Chameleon Imager”, CubeSatShop, 2020; https://www.cubesatshop.com/product/chameleon-imager/ (current as of Nov. 17, 2020).

[75] G. D. KREBS, “qbee50-LTU-OC (QB50 SE01)”, Gunter’s Space Page, 2020; https://space.skyrocket.de/doc_sdat/qbee.htm (current as of Nov. 17, 2020).

[76] “UHF Transceiver II”, EnduroSat, 2020; https://www.endurosat.com/cubesat-store/cubesat-communication-modules/uhf-transceiver-ii/ (current as of Nov. 17, 2020).

[77] “S-Band Receiver”, EnduroSat, 2020; https://www.endurosat.com/cubesat-store/cubesat-communication-modules/s-band-receiver/ (current as of Nov. 17, 2020).

[78] “Frequency and Wavelength Designations”, Department of Atmoshperic Sciences, College of the Environment, University of Washington; https://atmos.washington.edu/~houze/FrequencyAndWavelengthDesignations.html (current as of Nov. 17, 2020).

[79] “X-Band 4x4 Patch Array”, EnduroSat, 2020; https://www.endurosat.com/cubesat-store/all-cubesat-modules/x-band-4x4-patch-array/ (current as of Nov. 17, 2020).

[80] “3U CubeSat Platform”, Innovative Solutions in Space B.V., 2020; https://www.isispace.nl/product/3u-cubesat-platform/ (current as of Nov. 17, 2020).

[81] “Portsmouth Gaseous Diffusion Plant Virtual Museum”, Fluor BWXT Portsmouth for the Department of Energy; http://www.portsvirtualmuseum.org/index.html (current as of Nov. 17, 2020).

[82] “X-band Transmitter”, EnduroSat, 2020; https://www.endurosat.com/cubesat-store/cubesat-communication-modules/x-band-transmitter/ (current as of Nov. 17, 2020).

[83] “Kongsberg Satellite Services Ground Station Services”, Kongsberg Satellite Services AS; https://www.ksat.no/services/ground-station-services/ (current as of Nov. 17, 2020).

[84] “Where Is the International Space Station?”, Analytical Graphics, Inc, 2020; https://help.agi.com/stk/11.0.1/Content/training/WhereisISS.htm (current as of Nov. 17, 2020).

[85] “3U CubeSat Platform”, EnduroSat, 2020; https://www.endurosat.com/cubesat-store/all-cubesat-modules/3u-cubesat-platform/ (current as of Nov. 17, 2020).

[86] G. ANDERSON and T. YOUNG, “NASA Awards Launch Services Contract for Environmental Satellite Mission”, National Aeronautics and Space Administration, 2020; https://www.nasa.gov/press-release/nasa-awards-launch-services-contract-for-environmental-satellite-mission (current as of Nov. 17, 2020).

[87] National Research Council, *People and Pixels: Linking Remote Sensing and Socal Science: A Brief History of Remote Sensing Applications, with Emphasis on Landsat,* Chap. 2, pp. 28-36, National Academies Press, Washington, DC (1998).

[88] “Geostationary Operational Environmental Satellites – R Series Mission Overview”, National Oceanic and Atmospheric Administration and National Aeronautics and Space Administration; https://www.goes-r.gov/mission/mission.html (current as of Nov. 17, 2020).

[89] J. A. BENEDIKTSSON, J. CHANUSSOT, and W. M. MOON, “Advances in Very-High-Resolution Remote Sensing”, *Scanning the Issue, Proc. of the IEEE,* Mar. 2013, vol. 101, no. 3 (2013).

[90] R. J. LEE and S. L. STEELE, “Military Use of Satellite Communications, Remote Sensing, and Global Positioning Systems in the War on Terror”, *Journal of Air Law and Commerce* **79**, 1, 69 (2014).

[91] “Satellite Data: What Spatial Resolution is Enough?”, Earth Observing System, 2020; https://eos.com/blog/satellite-data-what-spatial-resolution-is-enough-for-you/ (current as of Nov. 17, 2020).

[92] “Newcomers Earth Observation Guide”, European Space Agency; https://business.esa.int/
newcomers-earth-observation-guide (current as of Nov. 17, 2020).

[93] “Tour of the Electromagnetic Spectrum Infrared Waves”, National Aeronautics and Space Administration, Science Mission Directorate, 2010; https://science.nasa.gov/ems/07_
infraredwaves (current as of Nov. 17, 2020).

[94] “Digital Thermal Imaging Camera”, Omega Engineering Inc., 2019; https://www.omega.com/en-us/test-inspection/thermal-imaging/p/TI-120-Series (current as of Nov. 17, 2020).

[95] K. J. HAVENS and E. J. SHARP, *Thermal Imaging Techniques to Survey and Monitor Animals in the Wild A Methodology,* Chap. 3, pp. 35-62, Elsevier Inc. (2016).

[96] “Infrared Thermometry Understanding and using the Infrared Thermometer”, Calex Electronics Limited; http://www.calex.co.uk (current as of Nov. 17, 2020).

[97] S. P. MAHULIKAR, G. A. RAO, H. R. SONAWANE, and H. S. S. PRASAD, “Infrared Signature Studies of Aircraft and Helicopters”, *Proc. of Progress In Electromagnetics Research symposium,* Moscow, Russia, Aug. 18-21, 2009, pp. 26-30 (2009).

[98] D. MANOLAKIS, S. GOLOWICH, and R. S. DIPIETRO, “Long-Wave Infrared Hyperspectral Remote Sensing of Chemical Clouds: A focus on signal processing approaches”, *IEEE Signal Processing Magazine* **31**, 4, 120-141, 12 Jun. 2014, IEEE (2014).

[99] E. A. MASON, “Gas”, Encyclopedia Britannica, 11 Nov. 2020; https://www.britannica.com/science/gas-state-of-matter (current as of Nov. 17, 2020).

[100] J. BRILL, “Ozone Monitoring Instrument (OMI)”, National Aeronautics and Space Administrations, Aura Atmospheric Chemistry; https://aura.gsfc.nasa.gov/omi.html (current as of Nov. 17, 2020).

[101] J. L. CULHANE, C. M. KORENDYKE, T. WATANABE, and G. A. DOSCHEK, “Extreme-Ultraviolet Imaging Spectrometer Designed for the Japanese Solar-B Satellite”, *Proc. of International Symposium on Optical Science and Technology,* San Diego, CA, 18 Dec. 2000, vol. 4139, SPIE (2000).

[102] “Panchromatic”, Earth Observing System, 2020; https://eos.com/panchromatic/ (current as of Nov. 17, 2020).

[103] “Remote Sensing Technology”, STARS project; https://www.stars-project.org/en/ (current as of Nov. 17, 2020).

[104] “GSP 216 Introduction to Remote Sensing Thermal Sensors and Satellites”, Humboldt State University Geospatial Online, 2019; http://gsp.humboldt.edu/OLM/Courses/ GSP_216_Online/lesson8-1/sensors.html (current as of Nov. 17, 2020).

[105] C. CLERBAUX, S. TURQUETY, and P. COHEUR, “Infrared Remote Sensing of Atmospheric Composition and Air Quality: Towards Operational Applications”, *Comptes Rendus Geoscience* **342**, 4-5, 349-356 (2010).

[106] “SkySat-C Generation Satellite Sensors”, Satellite Imaging Corporation, 2017; https://www.satimagingcorp.com/satellite-sensors/skysat-1/ (current as of Nov. 17, 2020).

[107] “DMC Constellation”, Airbus, 2020; https://www.intelligence-airbusds.com/en/8695-dmc-constellation (current as of Nov. 17, 2020).

[108] R. WRIGHT, P. LUCEY, L. FLYNN et al., “HYTI: Thermal Hyperspectral Imaging From a Cubesat Platform”, *Proc. of 33rd Annual AIAA/USU Conf. on Small Satellites*, SSC19-WKIV-02 (2019).

[109] A. PIRO, D. CASELLA, L. DI CIOLO, S. PINORI, et al., “HYBRIS: An Earth Observation Hyperspectral CubeSat Mission in Synergy with ESA Sentinels Missions”, (2017).

[110] J. A. FERNANDEZ-SALDIVAR and C. I. UNDERWOOD, “A Miniature UV Imaging Spectrometer for Remote Sensing of the Atmosphere”, presented at 22nd Annual AIAA/USU Conf. on Small Satellites, Logan, UT, 11-14 Aug. 2008.

[111] V. COTTINI, S. ASLAM, E. D’AVERSA et al., “CUVE – Cubesat UV Experiment: Unveil Venus’ UV Absorber with Cubesat UV Mapping Spectrometer”, *EPSC Abstracts* **11**, EPSC2017-771, 2017, European Planetary Science Congress (2017).

[112] D. W. PACK, D. R. ARDILA, E. HERMAN, D. W. ROWEN et al., “Two Aerospace Corporation CubeSat Remote Sensing Imagers: CUMULOS and R3”, *Proc. of 31st Annual AIAA/USU Conf. on Small Satellites,* SSC17-III-05 (2017).

[113] R. C. CARROLL, J. HAWKINS and D. THORSEN, "A Multispectral Thermal Imaging System for Measuring Volcanic Ash Mass Loading," *2015 IEEE Aerospace Conference*, Big Sky, MT, 2015, pp. 1-12 (2015).

[114] C. CLARK, K. VIERGEVER, A. VICK, and I. BRYSON, “Achieving Global Awareness via Advanced Remote Sensing Techniques on 3U CubeSats”, *Proc. of 26th Annual AIAA/USU Conf. on Small Satellites,* SSC12-IV-2 (2012).

[115] “EarthCARE Mission Instruments”, European Space Agency, 2020; https://earth.esa.int/web/guest/missions/esa-future-missions/earthcare (current as of Nov. 17, 2020).

[116] E. MAZARICO, G. A. NEUMANN, P. G. LUCEY, N. E. PETRO, X. SUN, and J. B. ABSHIRE, “Highe-Resolution LIDAR Spectroscopy for Solar System Exploration”, *Planetary Science Vision 2050 Workshop 2017*, LPI Contrib. no. 1989 (2017).

[117] M. STORM, H. CAO, D. ENGIN, and M. ALBERT, “CubeSat Lidar Concepts for Ranging, Topology, Sample Capture, Surface, and Atmospheric Science”, *Proc. of 31st Annual AIAA/USU Conf. on Small Satellites,* SSC17-S2-03 (2017).

[118] A. BEHRENDT, “Temperature Measurements with Lidar”, *Lidar*, pp. 273-305 (2016).

[119] E. PERAL, E. IM, L. WYE, S. LEE, S. TANELLI et al., “Radar Technologies for Earth Remote Sensing from CubeSat Platforms”, *Proc. of the IEEE*, Mar. 2018, vol. 106, no. 3 (2018).

[120] L. ACHEY, C. ALEXANDER, S. SHARMA, B. C. GUNTER, C. VALENTA, “The Development of a CubeSat Sized Imaging LiDAR”, Tethering and Ranging Mission of the Georgia Institute of Technology (2018).

[121] F. CHOLLET, *Deep Learning with Python,* Chap. 2-5, pp. 25-177, Manning Publications Co., Shelter Island, NY (2018).

[122] B. K. SPEARS, J. BRASE, P. BREMER, B. CHEN, J. FIELD et al., “Deep Learning: A Guide For Practitioners in the Physical Sciences”, *Phys. Plasmas* **25**, 080901 (2018).

[123] J. BROWNLEE, “Multi-Label Classification with Deep Learning”, Machine Learning Mastery Pty. Ltd, 2020; https://machinelearningmastery.com/multi-label-classification-with-deep-learning/ (current as of Nov. 17, 2020).

[124] R. SANTHOSHKUMAR and M. K. GEETHA, “Deep Learning Approach for Emotion Recognition from Human Body Movements with Feedforward Deep Convolution Neural Networks”, *Procedia Computer Science* **152**, 158-165 (2019).

[125] X. LI and S. WANG, "Object Detection Using Convolutional Neural Networks in a Coarse-to-Fine Manner," in *IEEE Geoscience and Remote Sensing Letters* **14**, 11, 2037-2041 (2017).

[126] S. IOFFE and C. SZEGEDY, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, *Proc. of the 32nd International Conference on Machine Learning,* Lille, France, 2015, W&CP vol. 37 (2015).

[127] D. GODOY, “Understanding Binary Cross-Entropy/Log Loss: A Visual Explanation”, Towards Data Science, 2018; https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a (current as of Nov. 17, 2020).

[128] G. HINTON, N. SRIVASTAVA, and K. SWERSKY, “Neural Networks for Machine Learning Lecture 6a Overview of Mini-Batch Gradient Descent”, University of Toronto, Computer Science (2014).

[129] S. RUDER, “An Overview of Gradient Descent Optimization Algorithms”, Sebastian Ruder, 2016; https://ruder.io/optimizing-gradient-descent/index.html#rmsprop (current as of Nov. 17, 2020).

[130] INL ART Program, “Key Regulatory Issues in Nuclear Microreactor Transport and Siting”, Idaho National Laboratory, US Department of Energy (2019).

[131] Global First Power, “Project Description for the Micro Modular Reactor Project at Chalk River”, CRP-LIC-01-001, 8 Jul. 2019, pp. 18-26 (2019).

[132] United States Nuclear Regulatory Commission, “Part 71 – Packaging and Transportation of Radioactive Material”, US NRC; https://www.nrc.gov/reading-rm/doc-collections/cfr/part071/full-text.html (current as of Nov. 17, 2020).

[133] R. B. POPE, “Packaging and Transport of Radioactive Material in the Nuclear Fuel Cycle”, *Nuclear Fuel Cycle Science and Engineering*, pp. 558-598, Woodhead Publishing (2012).

[134] “Uranium Radiation Properties”, WISE Uranium Project; https://www.wise-uranium.org/rup.html (current as of Nov. 17, 2020).

APPENDIX A

This appendix includes all the tables for the parameters of the fifteen common types of construction and mining equipment [35,36,37,38,39,40,41,42,43,44,45,46,47].

Table A1. The table lists the possible parameters for the identification of an excavator. The values come from the CAT 320 GC medium-sized excavator.

Dimension	Value
Height to Top of Cab	2.96 m
Length	9.53 m
Width	3.17 m
Track Length	3.27 m

Table A2. The table lists the possible parameters for the identification of a backhoe. The values come from the Bobcat B250 backhoe.

Dimension	Value
Height to Top of Cab	2.38 m
Load Height	2.41 m
Length	5.88 m
Width	1.62 m
Max Speed	6.44 km/h

Table A3. The table lists the possible parameters for the identification of a dragline excavator. The values come from the CAT 8000 dragline excavator.

Dimension	Value
Boom Length	101 m
Bucket Capacity	24-34 m^3

Table A4. The table lists the possible parameters for the identification of a bulldozer. The values come from the CAT D10 bulldozer.

Dimension	Value
Height to Top of Cab	4.54 m
Length with Blade	7.59 m
Width with Blade	3.51 m
Track Length	3.93 m

Table A5. The table lists the possible parameters for the identification of a grader. The values come from the CAT 120M grader.

Dimension	Value
Height to Top of Cab	3.29 m
Length	8.50 m
Width	2.50 m
Blade Base	2.53 m
Top Speed	44.57 km/h

Table A6. The table lists the possible parameters for the identification of a wheel tractor scraper. The values come from the CAT 637K wheel tractor scraper.

Dimension	Value
Height to Top of Cab	3.73 m
Length	15.48 m
Width	3.94 m
Top Speed	55.8 km/h

Table A7. The table lists the possible parameters for the identification of a trencher. The values come from the Wolfe 7000/7000D trencher.

Dimension	Value
Height to Top of Cab	3.55 m
Length	11.15 m
Width	3.48 m
Top Speed	4.75 km/h

Table A8. The table lists the possible parameters for the identification of a loader. The values come from the CAT 950M medium-sized loader.

Dimension	Value
Height to Top of Cab	3.45 m
Length without Bucket	6.91 m
Width	2.82 m
Bucket Capacity	9.20 m^3
Top Speed	39.5 km/h

Table A9. The table lists the possible parameters for the identification of a tower crane. The values come from the Manitowoc MDT 269 J10 tower crane.

Dimension	Value
Height	74 m (Highest crane in the world can reach up to 167.64 m)
Boom Length	65 m

Table A10. The table lists the possible parameters for the identification of a paver. The values come from the CAT BG240C paver.

Dimension	Value
Height to Top of Cab	2.77 m
Length	6.80 m
Width	2.44 m
Paving Speed	1.27 m/s
Paving Width	7.32 m
Top Speed	16.09 km/h

Table A11. The table lists the possible parameters for the identification of a compactor. The values come from the CAT 816K landfill compactor.

Dimension	Value
Height to Top of Cab	3.88 m
Length	8.06 m
Width	3.34 m
Blade Width	3.66 m
Top Speed	12.5 km/h

Table A12. The table lists the possible parameters for the identification of a telehandler. The values come from the JLG telehandler.

Dimension	Value
Height to Top of Cab	2.54 m
Length	6.12 m
Width	2.56 m
Maximum Lifting Height	16.76 m
Maximum Forward Reach	12.80 m
Top Speed	32.19 km/h

Table A13. The table lists the possible parameters for the identification of a feller buncher. The values come from the John Deere 643L-II feller buncher.

Dimension	Value
Height to Top of Cab	3.20 m
Length	6.27 m
Width	2.90 m

Table A14. The table lists the possible parameters for the identification of a dump truck. The values come from the CAT 7856 Tier 4 Final/Stage V dump truck.

Dimension	Value
Height to Top of Cab	5.21 m
Length	11.99 m
Outside Body Width	6.71 m
Top Speed	54.80 km/h

Table A15. The table lists the possible parameters for the identification of pile boring equipment. The values come from the CAT MD6250 drill.

Dimension	Value
Height with Mast Up	19.55 m
Height Mast Down	5.13 m
Body Length	11.71 m
Length Mast Down	20.18 m
Width	5.62 m
Top Speed	2.45 km/h

APPENDIX B

This appendix includes visual representations of the different pixel configurations for the phenomena of interest objects in the datasets created in Section 5. It is important to note that Figure 3B does not include every possible combination of pixels for an object of size 24 (Version A).

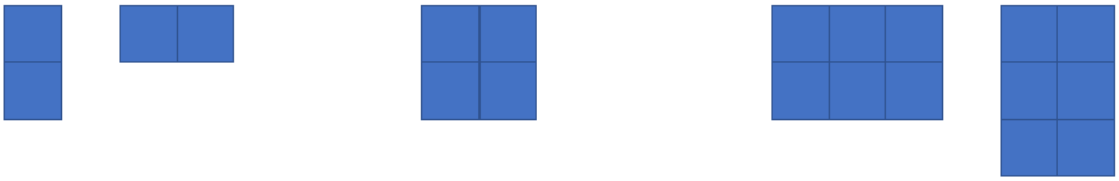


Figure B1. The different pixel configurations for objects with size 2, 4, and 6.

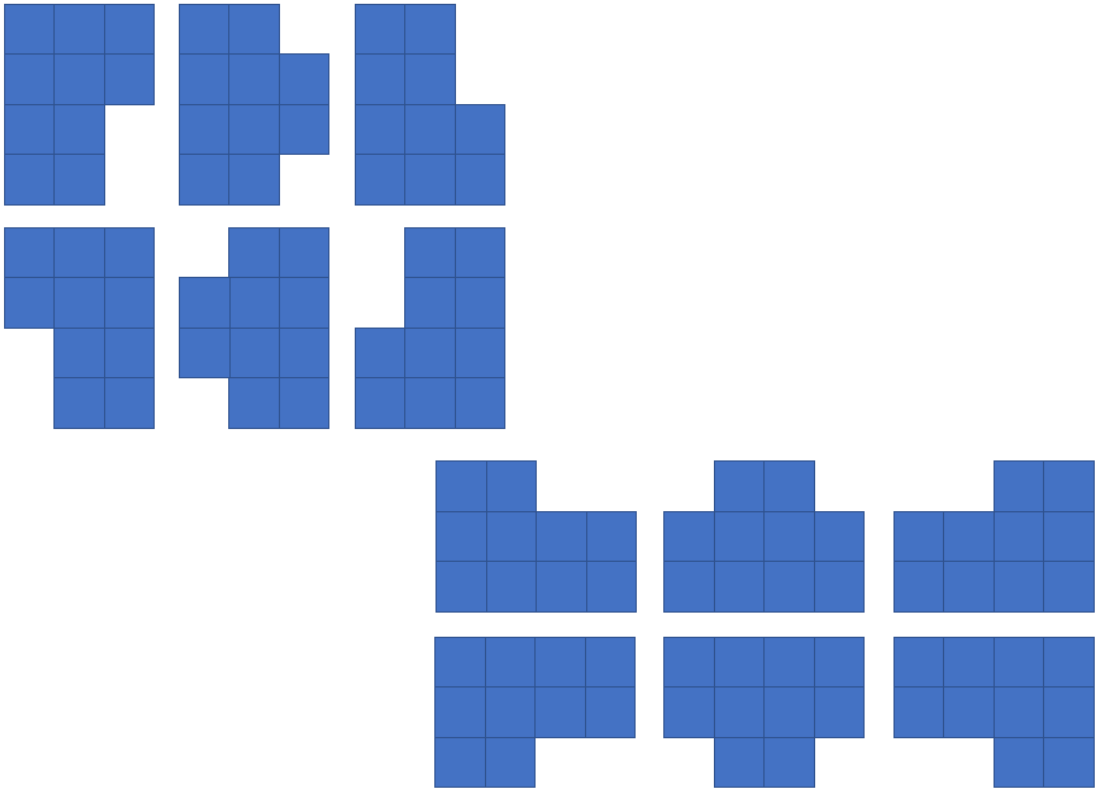


Figure B2. The different pixel configurations for objects with size 10.

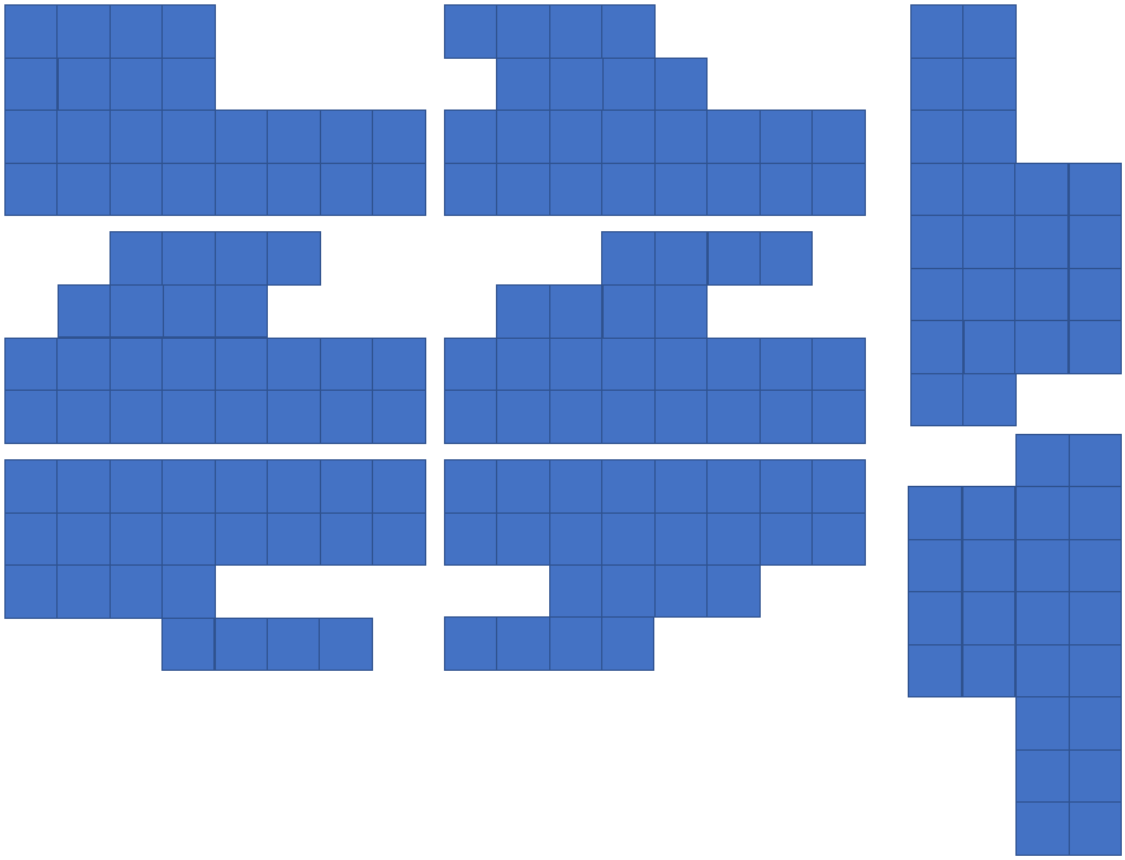


Figure B3. Some (not all) of the different pixel configurations for objects with size 24 (Version A).

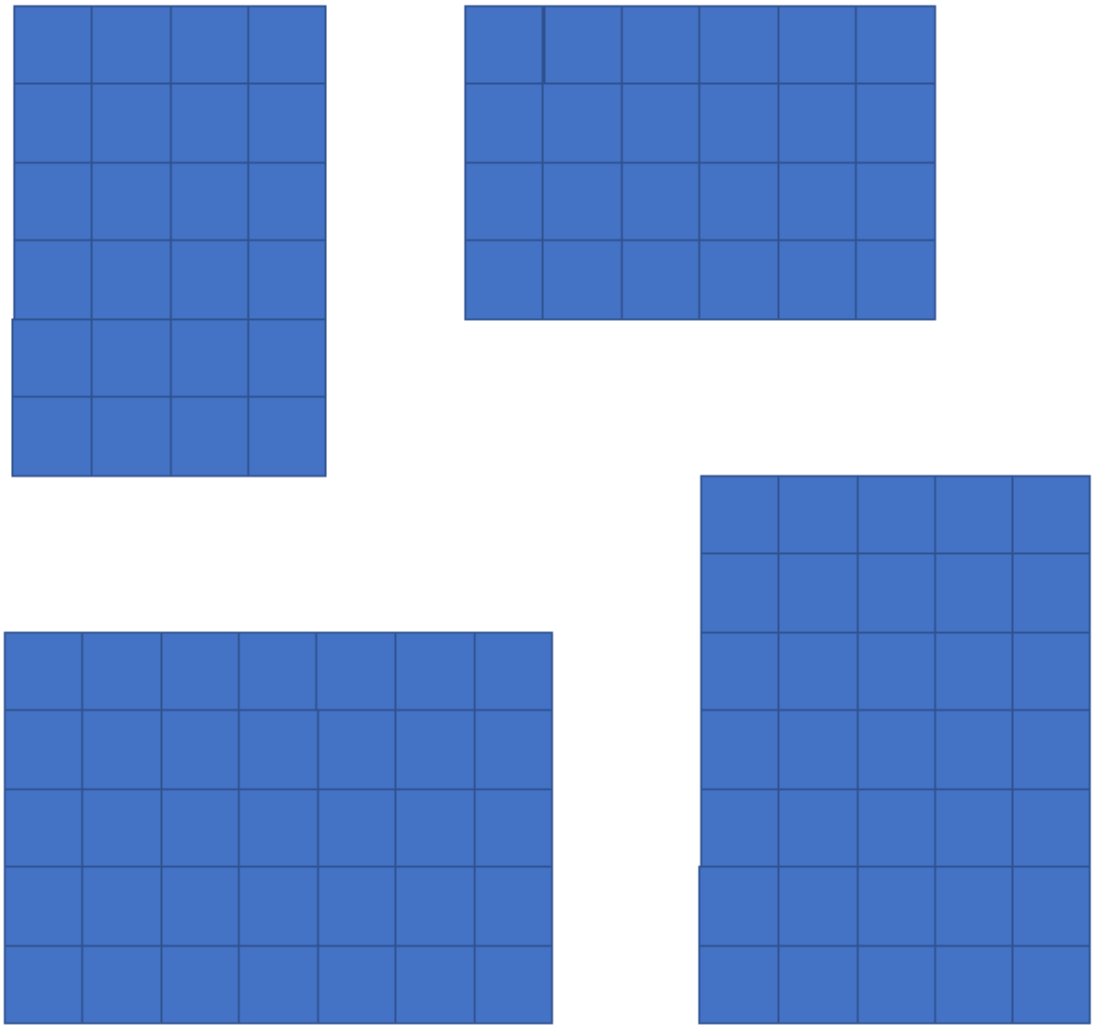


Figure B4. The different pixel configurations for objects with size 24 (Version B) and 35.