SMARTSAT
COOPERATIVE RESEARCH CENTRE

TECHNICAL REPORT 11

ENERGY-EFFICIENT ONBOARD AI FOR EARLY FIRE-SMOKE DETECTION

This report should be cited as:

SmartSat 2023, Energy efficient onboard AI for early fire-smoke detection, SmartSat Technical Report no. 11, SmartSat, Adelaide, Australia.

Acknowledgement:

SmartSat acknowledges the contribution made by Stefan Peters, Sha Lu, Eriita Jones, Jixue Liu, Jiuyong Li, Jim O’Hehir (University of South Australia; Kai Qin, Yu Sun (Swinburne University); Norman Mueller, Simon Oliver (Geoscience Australia).

Executive Summary

This report presents the findings of a research project undertaken by the University of South Australia (UniSA), Swinburne University of Technology (Swinburne), and Geoscience Australia (GA), under the leadership of UniSA. The study was conducted between March 2022 and October 2023 and involved 12 team members (UniSA: Stefan Peters, Sha Lu, Eriita Jones, Jixue Liu, Jiuyong Li, Jim O'Hehir; Swinburne: Kai Qin, Yu Sun; GA: Norman Mueller, Simon Oliver).

This research project, part of the Kanyini mission, aims to provide a solution for energy-efficient AI-based onboard processing of hyperspectral imagery for early fire smoke detection. It consists of two phases: pre-launch and post-launch. The pre-launch phase focuses on evaluating the feasibility and benefits of onboard smoke detection, while the post-launch phase deploys and optimizes the AI-based fire smoke detection system to Kanyini, leveraging real satellite data and refining onboard processing capabilities to enhance accuracy and efficiency. This report presents our work in the pre-launch phase. To evaluate the feasibility of onboard smoke detection, we employed a lightweight CNN-based fire smoke detection model, Variant Input Bands for Smoke Detection (VIB_SD) [12], which is capable of operating within the computational and data transfer constraints of the HyperScout-2 sensor on the Kanyini satellite. Since the actual computing environment and hyperspectral imagery of the Kanyini satellite were not available during the early stages of the mission, we simulated a comprehensive training dataset from VIIRS imagery, which encompassed a wide range of generated hyperspectral imagery, and the dataset covers various fire smoke scenarios across Australia.

To evaluate the performance of onboard image processing and AI tasks, we build an emulation system to simulate the computing environment of the Kanyini satellite. For the hardware, the emulation system uses a Raspberry Pi and an Intel Neural Compute Stick (NCS). We align the emulation system through software configurations to closely match the Kanyini satellite specifications. In terms of software, we have developed a pipeline that facilitates performance measurement, including running time, memory usage, and power consumption, for a wide range of onboard tasks. This work fills the gap of using a small model to detect fire smokes in a cube sat with limited computation and data down-linking capabilities. We report detailed hardware, software settings, and the performances of not-only smoke model accuracy, but also results about bands selection, and downlinking data sizes. Our experiments demonstrated that, comparing the emulation results of onboard and on-ground smoke detection scenarios, we found that AI onboard significantly reduced the data downlink volume to just 16% of its original size, resulting in a 69% decrease in energy consumption. Furthermore, AI onboard enabled the detection of fire smoke 500 times faster than AI on-ground. In summary, the major contributions of our work presented in this paper are:

Generation of a comprehensive hyperspectral training dataset: We created a training dataset using VIIRS imagery that encompasses diverse variations of fire smoke. The dataset includes four fire smoke scene-related classes, namely "Smoke", "Cloud", "Mixed", and "Clear". This dataset provides a valuable resource for training and evaluating fire smoke detection models.

Emulation system for onboard performance evaluation: We developed an emulation system to evaluate the onboard performance of different processes and AI tasks. This system serves as a valuable tool for optimizing algorithms and workflows before the deployment on actual satellite systems, ensuring the efficiency and effectiveness of onboard processing.

Deployment of the VIB_SD model: We adapted the VIB_SD model, a lightweight 88 CNN-based approach suitable to operate within the computational and data transfer 89 constraints of the HyperScout-2 sensor on the Kanyini satellite. The VIB_SD model 90 demonstrates a high prediction accuracy and achieves a low false negative rate on the 91 simulated dataset, indicating its effectiveness in detecting fire smoke.

Comprehensive experimental evaluation: We conducted comprehensive experiments to evaluate the various scenarios of onboard processing, providing empirical evidence on the feasibility and significance of onboard smoke detection. These experiments validate the benefits of onboard smoke detection in terms of downlink efficiency, energy consumption, and detection speed.

Although the emulation system is configured to match the Kanyini satellite, it is designed to be adaptable and easily applicable to other missions.

Table of Contents

Abbreviations

- Artificial intelligence (AI)
- Bands with low importance (LBI)
- Convolutional Neural Networks (CNNs)
- Eyes of Things (EOT)
- False negative rate (FNR)
- False positive rate (FPR)
- Flame And Smoke Detection Dataset (FASDD)
- Ground Truth (GT)
- HyperScout-2 (HS2)
- HyperScout-2 Smoke Detection Algorithm (HSSDA)
- Land use land cover (LULC)
- Near Infrared (NIR)
- Neural Compute Stick (NCS)
- Normalized Difference Red Blue (NDBR)
- Normalized Difference Vegetation Index (NDVI)
- Normalized Difference Violet NIR (NDVNIR)
- Onboard data handling system (ODHS)
- Open Visual Inference and Neural network Optimization (OpenVINO)
- Permutation Random Forest (PRF)
- Principal Component Analysis (PCA)
- Principal components (PCs)
- Random Forest (RF)
- Random Forest with Shapley value (RFS)
- Raspberry Pi (RPi)
- Ratio NIR to Blue (RNIRB)
- Red Edge (RedE)
- Region of interest (ROI)
- Shortwave infrared (SWIR)
- Smoke-Cloud spectral slope index 1 (SLOPE1)
- Smoke-Cloud spectral slope index 2 (SLOPE2)
- Thermal infrared (TIR)
- True negative rate (TNR)
- True positive rate (TPR)
- Ultraviolet (UV)
- Variant Input Bands for Smoke Detection (VIB_SD)
- Visible Infrared Imaging Radiometer Suite (VIIRS)
- Vision processing unit (VPU)
- Weighted Principal Component Analysis (wPCA)

1. Introduction

With the escalating threat of wildfires caused by climate change, advanced solutions for early fire detection are needed to minimize destructive impact of wildfire on ecosystems, societies, and economy [1, 2]. Satellite Remote Sensing has emerged as a cost-effective and reliable tool for fire detection, benefiting from the growing deployment of earth observation satellite systems [3-6]. As smoke is usually the first thing you can see from space before the fire gets hot and big enough for sensors to detect fire heat, detecting fire smoke becomes crucial for early warning and timely response to mitigate the potential risks and damages. However, the prevalence of microsats and nanosats and the increased spatial and spectral resolution of modern earth observation sensors have greatly increased bandwidth usage. This has led to research into innovative approaches to optimizing up/downlink bandwidth resources. For many sensor systems, only a fraction of the data collected contains mission-critical information related to the specific purpose of the mission. To address this issue, recent advances in low-power computing platforms and the advent of artificial intelligence (AI) technology have paved the way for the adoption of edge computing [7]. By leveraging hardware accelerators and deploying efficient algorithms like Convolutional Neural Networks (CNNs) onboard, tasks such as early fire smoke detection can be performed, allowing for timely alarm generation in the event of bushfires [8-10].

The Kanyini satellite mission [11] is a collaborative effort between the South Australian Government, the SmartSat Cooperative Research Centre, and local companies Inovor Technologies and Myriota. The mission aims to launch a 6U CubeSat satellite into low Earth orbit to collect data on bushfire preparedness, response and resilience, as well as inland 35 and coastal water quality. Equipped with a hyperspectral imager, the satellite sensor will capture reflected light from Earth in different wavelengths to generate detailed surface maps 37 for various applications, including bushfire monitoring, water quality assessment, and 38 land management. The anticipated launch year is 2023/2024, with an estimated cost of $6.539 million. The collected data will be publicly accessible and utilized by government agencies, businesses, and researchers. The Kanyini mission holds significance for South Australia's space 41 industry, being the first state-based satellite in Australia, and is expected to contribute to the state's space sector growth. Moreover, the acquired data will aid in enhancing bushfire and water resource management within South Australia.

This study seeks to develop energy-efficient AI onboard processing of hyperspectral imagery for the early detection of fire smoke. The project comprises two stages: pre-launch and post-launch. The pre-launch phase is focused on assessing the viability and advantages of onboard smoke detection. The post-launch phase involves deploying and refining the AI-based fire smoke detection system on Kanyini, utilizing actual satellite data to improve accuracy and efficiency in onboard processing. This report specifically outlines our efforts during the pre-launch phase.

To assess the feasibility of onboard smoke detection, we utilize a lightweight CNN-based model designed for fire smoke detection. This model operates within the computational and data transfer limits of the HyperScout-2 sensor on the Kanyini satellite. Due to the unavailability of the actual computing environment and hyperspectral imagery from the Kanyini satellite in the initial mission stages, we generated a comprehensive training dataset from VIIRS imagery. This dataset encompasses a diverse range of simulated hyperspectral imagery, covering various fire smoke scenarios across Australia.

The report is organized as follows: in Section 2, we provide an overview of the simulation process for the training dataset. In Section 3, we describe our onboard processing approach including emulation system architecture and design, Machine Learning based smoke detection model, implemented and tested onboard processing tasks, experiments and results. Section 4 presents the experiments and results on the prediction accuracy of VIB_SD with different selected bands and the emulation results of various onboard processes. Section 5 presents simulated satellite-ground station uplink and downlink data transfer investigations. Section 6 summarizes the key conclusions drawn from the study and provides insight into planned future research.

1.1. Related Work

The field of AI onboard satellite systems has witnessed significant progress, revolutionizing satellite operations and data analysis. Numerous works have explored the integration of AI algorithms and machine learning techniques into satellites, enabling autonomous decision-making, improved data processing, and enhanced mission performance. Various applications in satellite observation have been explored, including land cover classification [15, 16], vegetation monitoring [17, 18], water quality assessment [19], disaster response [20, 21], and climate change monitoring [22, 23]. One notable study in [13, 14] presents the first in-orbit demonstration of AI applied to hyperspectral and thermal sensing using the HyperScout-2 imager. Deployed in the ϕ-Sat-1 mission, it utilized a deep neural network for cloud detection with an impressive 95% accuracy, surpassing traditional cloud screening methods with around 80% accuracy. Similarly, another work [13] showcases the in-orbit demonstration of HyperScout-1, the first hyperspectral imager for nanosatellites, with potential applications in land cover classification, vegetation monitoring, water quality assessment, and disaster response. The FMPL-2 instrument's successful performance in orbit is reported in the works [24, 25]. The GNSS-R receiver accurately detected signals from GPS, GLONASS, and Galileo satellites, estimating sea ice concentration with 95% accuracy. The L-band microwave radiometer measured Earth's surface brightness temperature and estimated soil moisture with 10% accuracy based on ground-truth validation. Furthermore, the research in [10] proposes the use of a deep learning algorithm to segment images from low-cost satellites into different land cover classes, including water, for flood mapping. Their experiment using Sentinel-2 satellite images achieved an 85.6% accuracy in flood mapping using the proposed approach. These works collectively demonstrate a substantial progress in AI applications for satellite systems, enabling advanced data processing and critical applications in various domains such as environmental monitoring and disaster response.

To the best of our knowledge, the work [13, 14] is the only one similar in settings to ours but it is in the area of cloud detection. Our work is different in the following aspects: Our pre-launch study does not have available hyperspectral dataset matching the bandwidths of the sensor to be launched. Our work is for newer generation of VPUs and the development environment has been updated and some functions available in the version used in [13] are not available in the new version.

2. VIIRS-based simulated AI training dataset

To simulate HyperScout-2 imagery that will be captured by the upcoming Kanyini satellite mission, it was decided to modify VIIRS imagery over existing historical fire events for creating a simulated dataset used for model training and validation.

In this chapter the following is presented:

- Description of the VIIRS based simulation of HyperScout-2 imagery at 75 m/pixel resolution containing 48 spectral bands;
- Introducing a novel pixel- and spectral indices based deterministic fire smoke detection algorithm (HSSDA: HyperScout-2 Smoke Detection Algorithm) used a) as an alternative for the AI-based (VIB_SD) smoke detection method, and b) for the automatic creation of training and validation data;
- Creation of a comprehensive 256x256 pixels grid tile-based training dataset using VIIRS imagery that encompasses diverse variations of fire smoke. The dataset includes four fire smoke scene-related classes, namely "Smoke," "Cloud", "Mixed," and "Clear." This dataset provides a valuable resource for training and evaluating fire smoke detection models;
- Creation of an additional manual labelled training dataset.

2.1. Simulating HyperScout-2 Imagery

Since the Kanyini satellite is yet to be launched, one goal of this project was the generation of a pseudo imagery dataset at HyperScout-2 spectral and spatial resolution as it will be captured via the Kanyini satellite. The purpose of this is so that the AI onboard smoke detection model could be trained and developed on imagery that simulates both the information and technical properties of the datasets to be captured by the Kanyini satellite during flight. HyperScout-2 (HS2) is a hyperspectral imager with 45 spectral bands across the 400-1250 nanometre range and 3 bands at thermal wavelengths between 8-14 µm[1]. The key resolutions of HS2 are given below in Table 1. Initially MODIS Terra was being used in this project as a source of multispectral satellite imagery. However, as MODIS Terra was due to be decommissioned in 2022, the NASA/NOAA Suomi NPP satellite carrying the Visible Infrared Imaging Radiometer Suite (VIIRS) was chosen therefore to provide the imagery dataset used for hyperspectral imagery simulation instead. A comparison between the resolutions of VIIRS and HS2 is given in Table 1. VIIRS captures daily imagery over the Australian continent, however it has a coarser resolution of at best 375 m for visible – shortwave infrared (SWIR), and 1000 m for thermal infrared (TIR)[2]. It also has a much coarser spectral resolution being a multi- not hyperspectral camera, and it has only 11 bands that provide a wavelength overlap to those of HyperScout-2. Out of 48 spectral bands, this then leaves 27 HS2 spectral bands that required simulation. VIIRS' lower resolutions compared to HS2 do not impact our investigations of AI onboard processing performance testing using VIIRS-based simulated HS2 imagery dataset.

[1] Technical Kanyini/HyperScout-2 details provided by Nick Manser, Satellite Systems Engineer, SmartSat-CRC

[2] https://lpdaac.usgs.gov/data/get-started-data/collection-overview/missions/s-npp-nasa-viirs-overview/

TABLE 1: RESOLUTIONS OF HYPERSCOUT-2 AND VIIRS SENSORS.

	HyperScout-2 Wavelength (µm) [Band]	VIIRS Wavelength (µm) [Band]	Spectrum
Radiometric	16bit	12bit	
Spatial	**75 m** [visible – SWIR] 390 m [TIR]	375 m (I), 500 m (M) [visible – SWIR] 1000 m [TIR]	
Spectral	0.400-0.416 [B1]	0.402 - 0.422 [M1]	Violet
	0.432-0.448 [B3]	0.436 - 0.454 [M2]	Violet
	0.480-0.496 [B6]	0.478 - 0.488 [M3]	Blue
	0.544-0.560 [B10]	0.545 - 0.565 [M4]	Green
	0.640-0.656 [B16]	0.60 - 0.68 [I1]	Red
	0.672-0.688 [B18]	0.662 - 0.682 [M5]	Red
	0.736-0.752 [B22]	0.739 - 0.754 [M6]	Red-Edge
	0.848-0.864 [B29]	0.85 - 0.88 [I2]	NIR
	0.864-0.880 [B30]	0.846 - 0.885 [M7]	NIR
	10.05-11.24 [T2]	10.26 - 11.26 [M15]	TIR
	11.40-12.50 [T3]	11.54 - 12.49 [M16]	TIR

The VIIIRS imagery products utilised for simulating HyperScout-2 imagery are as follows:

- VNP09GA[3] - VIIRS/NPP Surface Reflectance Daily L2G Global 1 km and 500 m SIN Grid
- This is a surface reflectance product that provides an estimate of surface reflectance in the visible-SWIR bands I1-I2 and M1-M7. The observations are projected onto a two-dimensional grid and stored as 10 km square tiles at 500 m and 1 km resolution. Surface reflectance is obtained by adjusting top-of-atmosphere reflectance to compensate for atmospheric effects. Corrections are made for the effects of molecular gases, including ozone and water vapor, and for the effects of atmospheric aerosols. The

[3] https://ladsweb.modaps.eosdis.nasa.gov/missions-and-measurements/products/VNP09GA#overview

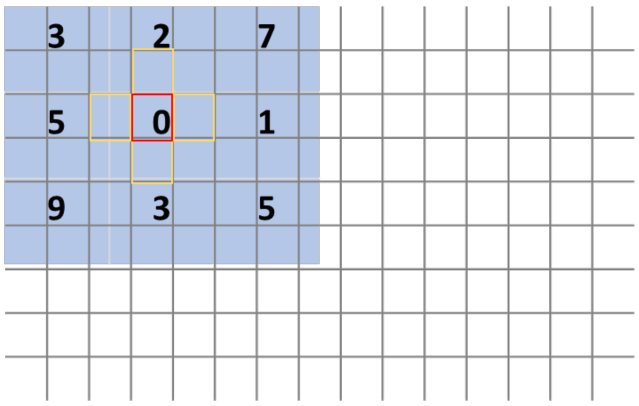
inputs to the surface reflectance algorithm include the VIIRS cloud mask and aerosol product. All surface reflectance products are produced for daytime conditions only. The product is produced under all atmospheric conditions except for night and oceans.

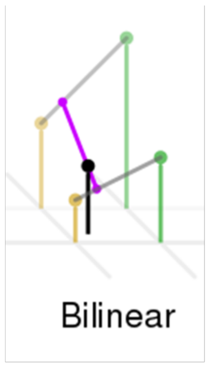
- VNP09CMG[4] - VIIRS/NPP Surface Reflectance Daily L3 Global 0.05 Deg CMG
- This product includes the moderate resolution surface emission and brightness temperature in the thermal bands M15-M16. The data are corrected for atmospheric conditions such as the effects of molecular gases, including ozone and water vapor, and for the effects of atmospheric aerosols. This product uses a weighted average of the best quality observation and is formatted as a climate modelling grid (CMG) for use in climate simulation models. In addition to the thermal data this product includes information layers representing relative azimuth angle, sensor zenith angle, solar zenith angle, reflectance band quality, time of day, and number mapping.

The VIIRS reflectance imagery (visible-SWIR wavelengths) was accessed via the Land Processes Distributed Active Archive Center (LP DAAC) APPEEARS data access tool[5], by uploading a polygon shapefile outlining the region of interest (ROI). The VIIRS emission imagery (thermal wavelengths) was accessed via direct download link available from LP DAAC. Once the satellite imagery data needed for each ROI had been accessed, HyperScout-2 simulated images were produced by first super-resolving the VIIRS imagery to the best expected resolution of HS2 image resolution. HS2 achieves a ground-sampling distance of 75 m/pixel at an orbit of 500 m altitude [26] (Kanyini is currently targeting a highly orbital altitude of approximately 550 km[6]). To super-resolve the imagery a simple bi-linear interpolation algorithm was used. This algorithm does not infer any new spatial information, but merely resamples the image pixel grid to a higher resolution, as illustrated in the schematic in Figure 2-1. The image on the left shows a portion of the original coarser resolution VIIRS image raster (behind, blue) and a portion of the new finer resolution HS2 image raster (the outlined grid). The value in the red cell is calculated from bilinear interpolation of the values of its four neighbouring cells in yellow. their values are obtained from sampling the nearest cell in the original VIIRS image. In this example, the neighbouring cells may have value (0,5) across the x direction and (0,2) across the y direction.

The second step in producing a simulated HS2 image was to fill the band gaps or missing spectral information. As discussed above, only 11 spectral bands of VIIRS directly overlapped those of HS2, leaving 27 HS2 bands to be simulated. To create these pseudo bands a combination of: (i) duplicating spectral information, where broader wavelength range VIIRS bands encompassed 2 or more HS2 bands; and (ii) spectrally averaging 2 VIIRS bands at a longer and shorter wavelength than the missing wavelength band, was applied. Poissonian distributed random noise R(λ) was multiplicatively added to each pseudo-band as appropriate for electromagnetic imagery (e.g. photon noise, [27]). The schema used to create the simulated HS2 wavelength information is given in Table 2.

[4] https://lpdaac.usgs.gov/products/vnp09cmgv001/
[5] https://lpdaac.usgs.gov/tools/appeears/
[6] Technical Kanyini/HyperScout-2 details provided by Nick Manser, Satellite Systems Engineer, SmartSat-CRC.





$$cell\ value = (a + bx) \times (x + dy)$$

FIGURE 2-1: ILLUSTRATION OF HOW BI-LINEAR INTERPOLATION RESAMPLES A RASTER TO PRODUCE A SUPER-RESOLVING IMAGE FROM 375-1000 M/PIXEL TO 75M/PIXEL.

TABLE 2: SOURCE OF SIMULATED HYPERSCOUT-2 WAVELENGTH INFORMATION.

HyperScout-2 Band	Central Wavelength[7]	Overlapping VIIRS Band	Formula
1	0.408	M1 Violet	M1 Violet
2	0.424	M1 Violet	(M1 Violet+M2 Violet)/2) × R(λ)
3	0.44	M2 Violet	M2 Violet
4	0.456	M2 Violet	(M3 Blue+M2 Violet)/2 × R(λ)
5	0.472	M3 Blue	(M3 Blue) × R(λ)
6	0.488	M3 Blue	M3 Blue
7	0.504	M3 Blue	(M3 Blue+M4 Green)/2 × R(λ)
8	0.52	M4 Green	(M4 Green) × R(λ)
9	0.536	M4 Green	(M4 Green) × R(λ)
10	0.552	M4 Green	(M4 Green) × R(λ)
11	0.568	M4 Green	(M4 Green) × R(λ)

[7] Technical Kanyini/HyperScout-2 details provided by Nick Manser, Satellite Systems Engineer, SmartSat-CRC.

12	0.584	M4 Green	(I1 Red+M4 Green)/2 × R(λ)
13	0.6	I1 Red	(I1 Red) × R(λ)
14	0.616	I1 Red	(I1 Red) × R(λ)
15	0.632	I1 Red	(I1 Red) × R(λ)
16	0.648	I1 Red	I1Red
17	0.664	M5 Red	(I1 Red+M5 Red)/2 × R(λ)
18	0.68	M5 Red	M5Red
19	0.696	M6 RedE	(M6 RedE+M5 Red)/2 × R(λ)
20	0.712	M6 RedE	(M6 RedE) × R(λ)
21	0.728	M6 RedE	(M6 RedE) × R(λ)
22	0.744	M6 RedE	M6 RedE
23	0.76	M6 RedE	(M6 RedE) × R(λ)
24	0.776	M6 RedE	(M6 RedE) × R(λ)
25	0.792	M6 RedE	(M6 RedE+I2 NIR)/2 × R(λ)
26	0.808	I2 NIR	(I2 NIR) × R(λ)
27	0.824	I2 NIR	(I2 NIR) × R(λ)
28	0.84	I2 NIR	(I2 NIR) × R(λ)
29	0.856	I2 NIR	I2 NIR
30	0.872	M7 NIR	M7 NIR
31	0.888	M7 NIR	(M7 NIR) × R(λ)

32	0.904	M7 NIR	(M7 NIR) × R(λ)
33	0.92	M7 NIR	(M7 NIR) × R(λ)
34	0.936	M7 NIR	(M7 NIR) × R(λ)
35	0.952	M7 NIR	(M7 NIR) × R(λ)
36	0.968	M7 NIR	(M7 NIR) × R(λ)
37	0.984	M7 NIR	(M7 NIR) × R(λ)
38	1	M7 NIR	(M7 NIR) × R(λ)
39	1.016	M7 NIR	(M7 NIR) × R(λ)
40	1.032	M7 NIR	(M7 NIR) × R(λ)
41	1.048	M7 NIR	(M7 NIR) × R(λ)
42	1.064	M7 NIR	(M7 NIR) × R(λ)
43	1.08	M7 NIR	(M7 NIR) × R(λ)
44	1.096	M7 NIR	(M7 NIR) × R(λ)
45	1.256	M7 NIR	(M7 NIR) × R(λ)
46	11	M15 THERM	(M15 Therm+M16 Therm)/2 × R(λ)
47	10.645	M15 THERM	M15 Therm
48	11.95	M16 THERM	M16 Therm

2.2. “HyperScout-2 Smoke Detection Algorithm” (HSSDA)

The “HyperScout-2 Smoke Detection Algorithm” currently consists of two primary codes, one written in the IDL programming language, which can be run through the IDL/ENVI software package produced by Harris[8], and one written in python and run through ArcGIS Pro. The primary code is in IDL, and exists as a series of .pro files, a master file which is the one in which the user enters their key inputs and then runs, and a series a sub files which are called and run automatically but the master file (all codes must be complied first). All .pro files can be

[8] https://www.l3harrisgeospatial.com/Software-Technology/IDL

saved and run from any directory once the IDL/ENVI software package is installed on the user's system, or they have access to a license via VPN. Table 3 shows the .pro files required for running HSSDA. The primary view when opening HSSDA_master_HS00.pro is shown in Figure 2-2. At one junction the user exists IDL to run a single code in ArcGIS Pro, and then returns to running the IDL master program. This is outlined in Table 3 below. The function of the HSSDA is described in detail in the text in the sections below.

TABLE 3: "HYPERSCOUT-2 SMOKE DETECTION ALGORITHM" *(HSSDA) PROCEDURES.*

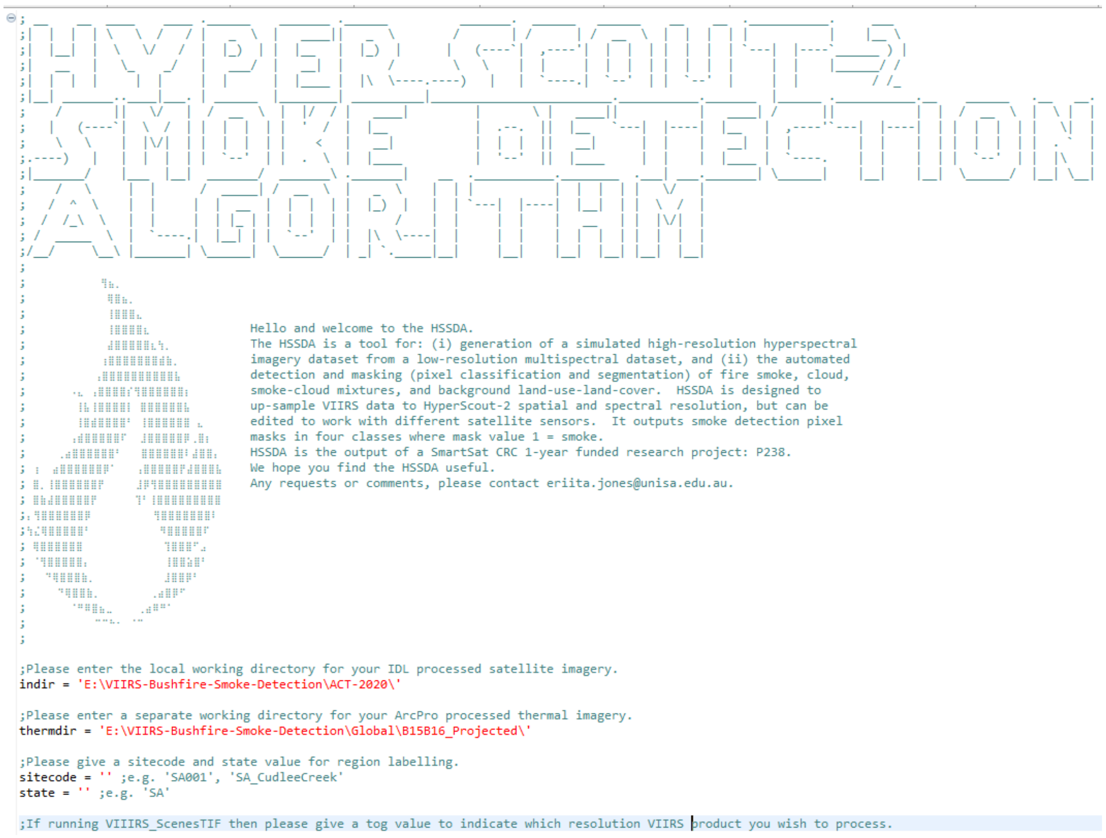
Filename	Primary Function
HSSDA_master_HS00.pro	Master file where user inputs directories of VIIRS imagery, input geospatial files, and a unique state and sitecode for labelling of output region.
VIIRS_ScenesTIF_HS01.pro	Reads in the downloaded .netcdf files from APPEARS containing *VNP09GA* visible to SWIR wavelength data, extracts the required bands of interests, and exports them as geotiff's in WGS 1984 projection.
VIIRS_StackVNIR_HS02.pro	Inputs the 500 m and 1000 m spatial resolution VIIRS outputs from the previous procedure and stacks them together to a single layered geotiff that is co-registered to the same spatial grid as the 500m product in a WGS 1984 projection, with the coarser spatial product resampled through the nearest neighbour method (no interpolation).
VIIRS_SmokeMask_HS03.pro	Inputs the 9 band multispectral image from the previous procedure and applies the HSSDA algorithm to assign pixels to aerosol classes and export the four class HSSDA raster mask described in the text.
VIIRS_RenameTherm_H04.pro	Reads in the downloaded .h5 files from LPDAAC containing *VNP09CMG* thermal brightness temperature data, converts the filename to a julian calendar date system, and exports the file with a new filename including the year-month-day for ease of matching with the corresponding *VNP09GA* products.
Here the HSSDA master program is temporary exited to run the following code in ArcGIS pro.	
ArcPro_extract_reproject.aprx	Extracts the two brightness temperature bands from the VIIRS "VNP09CMG" data products, reprojects them to WGS 1984 (WKID:4326) and outputs them as geotiffs.
Here the user returns to the HSSDA master program in IDL.	
VIIRS_StackTHERM_HS05.pro	Inputs the 9 band multispectral layerstack from procedure HS02, the two thermal bands projected by the python script in ArcPro, and the HSSDA mask. Stacks the 12 bands together, co-registers them onto the same spatial grid as the 500m product in a WGS 1984 projection, with the coarser thermal product resampled through the nearest neighbour method, and subset to the same

	spatial extent. The 12 spectral band image is then exported as a geotiff.
VIIRS_WaterMask_HS06.pro	Inputs the 12 band multispectral layerstacks with HSSDA mask from procedure HS05, and the “land surface mask” for the corresponding state (described in text below). The mask is applied to all spectral bands in every image. Pixels that are not retained (ie. those that correspond to water) are given a “no date” value). The masked multispectral images are exported as geotiffs.
VIIRS_HS2_Rebin_HS07.pro	Inputs the water-removed 12 band multispectral images from the previous procedure and super-resolves them to HyperScout-2 spatial resolution of 75 m/pixel through the method of bilinear interpolation, as described in the text. Resampled images are exported as geotiffs.
HS2_BandFilling_HS08.pro	Inputs the 12 band multispectral images that are now at the spatial resolution of HyperScout-2, and were the output of the previous procedure. Generates a 49 band floating point array where the last 48 bands are calculated from the input multispectral data according to the information given in Table 2, resulting in a simulated hyper-spectral image. The first band corresponds to the HSSDA mask which is here re-calculated, according to the formula given in the text due to the change of pixel values occurring with the bilinear interpolation. The new pseudo-HyperScout-2 images with HSSDA masks are then exported as geotiffs.
HS2_Tiling_HS09.pro	Every simulated HyperScout-2 image from the previous step is input and tiled according to the tile schematics given in the text. Each tile is only saved to disk if every pixel contains valid spectral data (ie. no image boundary pixels). Tiles are saved as 49 band geotiffs.
HS2_Tiling_Summarise_HS10.pro	Reads in all tiles by user given region and exports a text file containing a summary of the proportion of pixels in each tile that are assigned to each of the four HSSDA classes (example shown in Figure 2-7).
HS2_Tiling_NetCDF_HS11.pro	In this optional procedure, each tile exported by procedure HS09 is here input and exported as a .netcdf file with attribute descriptions for each of the 49 HSSDA mass and spectral bands.

All in table listed HSSDA files are available via:

https://github.com/eriita/SmartSatCRC_P238_HSSDA

(*access can be provided on request*)



```
Hello and welcome to the HSSDA.
The HSSDA is a tool for: (i) generation of a simulated high-resolution hyperspectral
imagery dataset from a low-resolution multispectral dataset, and (ii) the automated
detection and masking (pixel classification and segmentation) of fire smoke, cloud,
smoke-cloud mixtures, and background land-use-land-cover.  HSSDA is designed to
up-sample VIIRS data to HyperScout-2 spatial and spectral resolution, but can be
edited to work with different satellite sensors.  It outputs smoke detection pixel
masks in four classes where mask value 1 = smoke.
HSSDA is the output of a SmartSat CRC 1-year funded research project: P238.
We hope you find the HSSDA useful.
Any requests or comments, please contact eriita.jones@unisa.edu.au.

;Please enter the local working directory for your IDL processed satellite imagery.
indir = 'E:\VIIRS-Bushfire-Smoke-Detection\ACT-2020\'

;Please enter a separate working directory for your ArcPro processed thermal imagery.
thermdir = 'E:\VIIRS-Bushfire-Smoke-Detection\Global\B15B16_Projected\'

;Please give a sitecode and state value for region labelling.
sitecode = '' ;e.g. 'SA001', 'SA_CudleeCreek'
state = '' ;e.g. 'SA'

;If running VIIIRS_ScenesTIF then please give a tog value to indicate which resolution VIIRS product you wish to process.
```

FIGURE 2-2: SCREENSHOT OF HSSDA PROGRAM.

2.3. Deriving Smoke and Cloud Aerosol Masks

The second remote sensing goal of this project was to derive a method for the automated detection of smoke, and its differentiation from other aerosols such as cloud, dust, air pollution, etc. This method would then be utilized to derive a per-pixel mask for each simulated hyperspectral image where every pixel in the image would be assigned a value classifying it into one of four classes according to the following system, where the pixel mask value is equal to the class value (ie. if a pixel is assigned to class 1 then in the mask band the pixel will have value 1):

- Class 0 ⇨ “no aerosol” (ie. no fire smoke, no haze, no cloud, etc).
- Class 1 ⇨ “smoke endmembers” (ie. strongest smoke signatures).
- Class 2 ⇨ “aerosol mixels” (ie. mixtures of fire smoke, with cloud, other aerosols).
- Class 3 ⇨ “cloud endmembers” (ie. strongest cloud signatures).

The purpose of this deterministic (spectral-threshold based) smoke detection method is (a) to automatically derive a large number of training samples for our AI onboard processing performance testing, and (b) to apply this deterministic smoke detection method onboard as an alternative approach, or pre-processing step to our AI onboard smoke detection. Note that the implementation of (b) is not part of SS-p2.38 but foreseen for the next stage of this research (post-launch phase).

To detect clouds in satellite imagery, visible wavelengths are typically used as clouds are amongst the brightest features in an image. The detection of cloud shadows is also crucial as final cloud detection requires both cloud and cloud shadow probability above a crucial threshold [28] (e.g. the FMASK algorithm). Shadow detection incorporates thermal wavelengths, which are acquired by HS2, however for sensors without thermal bands other methods of cloud-shadow spatial relationships are possible, and additional SWIR wavelength information at 1.4 µm, which captures water absorption features, assists in cloud detection [29] (ie. Sentinel-2's "cirrus" band, this information is not captured by HS2). Visible and near infrared wavelengths are insufficient to discriminate smoke from cloud. Bushfire smoke can also be bright and white in an image, particularly in a low temperature burn, high fuel moisture scenario making the spectral discrimination of smoke from cloud across visible and NIR wavelengths challenging, as shown in Figure 2-3. Hence other wavelengths become extremely useful for capturing the unique particulate distributions and chemical properties of both aerosols.

Smoke plume detection is frequently undertaken using aerosol indices which incorporate ultraviolet wavelengths to capture the spectral properties of smoke particles [30]. HyperScout-2 however will not acquire information in the UV. Smoke and cloud differentiation can be assisted by incorporating longer wavelength information in the shortwave infrared regime, captured between wavelengths of 1.6-2.2 µm (e.g. see Zhao et al. [12]). Both Landsat 8 and Sentinel 2 have 2 SWIR bands in this spectral range, however HS2 will not acquire spectral information in these wavelength regions. Due to limited information to be acquired at SWIR wavelengths, and no data being acquired in the UV, we developed new methods for smoke detection and smoke cloud differentiation.

All of the aforementioned smoke and cloud detection approaches become less effective when the smoke plume or cloud is not opaque. Smoke and cloud can spectrally overlap when they are thin and partially transparent to the satellite sensor. In cloud or pixel quality masking products generated alongside satellite imagery, semitransparent cloud may be defined as optically thin cirrus cloud, thin lower altitude cloud, haze or indeed smoke [12]. This is expected to also provide a limiting factor for our algorithm but may be partially mitigated once additional spatial measures of smoke plume shape, cloud shadow, and feature texture are incorporated.

The colour of smoke can indicate the fire's behaviour.

Smoke colour	Indicates
Dense, white	Very moist fuels, mild behaviour
Pale grey/blue	Moist fuels, mild-moderate fire
Black/dark brown	Dry fuels, high fire behaviour
Copper/bronze	Very dry fuels, high to severe fire behaviour
Smoke column	**Indicates**
Thin, lazily rising	Small fire with low speed
Bent over close to ground and increasing in volume	Wind-driven fire that is speeding rapidly

Blue smoke – low-intensity fire

Dark brown smoke – high intensity
Photo: Anna Blaney

Smoke column bent close to the ground and increasing in volume – wind-driven fire that is spreading rapidly

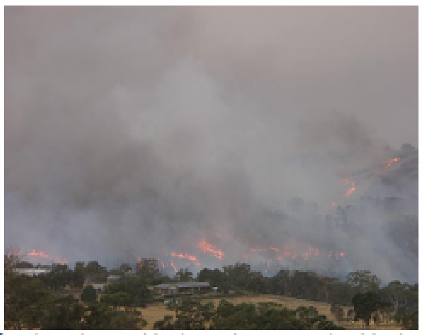

Smoke column widening at base, mostly white but turning brown on downside – fire is spreading in grass and moving into heavier fuels
Photo: Mari Lawson

FIGURE 2-3: THIS FIGURE "SMOKE AS AN INDICATOR" TAKEN FROM CFS FACT SHEET: BUSHFIRE BEHAVIOUR IN DETAIL, PRODUCED BY THE GOVERNMENT OF SOUTH AUSTRALIA, #6, OCTOBER 2010.

The general approach for developing the "HSSDA" or "HyperScout-2 Smoke Detection Algorithm" here was a risk adverse approach with the framework adapted broadly from [31] and refined via empirical research. In this approach, pixels which were highly likely to not belong to one of the aerosol classes of interest (class 1-3) were removed. These pixels belong to background land-use and land-cover (LULC) categories such as vegetation, soil, urban and other bright features. The pixels that remained where then classified by a successive masking approach.

The formula for the "HyperScout-2 Smoke Detection Algorithm" (HSSDA), and the method for assigning pixels to classes, is as follows:

- If (NDVI $\geq$ 0.51) or (NDBR $\geq$ 0.2) or (RNIRB $\geq$ 4.5) or (NDVNIR $\geq$ 0.31) then:
  - Pixel ⇨ mask value = 0, Class 0, “no aerosol”
- If (SLOPE2 < 5000) and (HS2_B16[1]) < 1500 then:
  - Pixel ⇨ mask value = 1, Class 1, “smoke endmembers”
- If (HS2_B16) > 3000 then:
  - Pixel ⇨ mask value = 3, Class 3, “cloud endmembers”
- Otherwise:
  - Pixel ⇨ mask value = 2, Class 2, “aerosol mixels”

Where the spectral indices are defined below, both in general wavelength regions, and in comparison to HS2 band numbers:

- Normalized Difference Vegetation Index (NDVI)
  - NDVI = (NIR – RED)/(NIR + RED) = (HS2_B30 – HS2_B16)/ (HS2_B30 + HS2_B16)
- Normalized Difference Red Blue (NDBR)
  - NDBR = (RED - BLUE)/(RED + BLUE) = (HS2_B16 – HS2_B6)/ (HS2_B16 + HS2_B6)
- Ratio NIR to Blue (RNIRB)
  - RNIRB = NIR / BLUE = HS2_B30/HS2_B6
- Normalized Difference Violet NIR (NDVNIR)
  - NDVNIR = (VIOLET - NIR)/(VIOLET + NIR) = (HS2_B1 – HS2_B29)/ (HS2_B1 + HS2_B29)
- Smoke-Cloud spectral slope index 1 (SLOPE1)
  - SLOPE1 = function of REDE and Red (see Appendix B.1, confidential)
- Smoke-Cloud spectral slope index 2 (SLOPE2)
  - SLOPE2 = function of NIR and REDE (see Appendix B.1, confidential)

The SLOPE1 and SLOPE2 indices were developed here as they were found to differentiate key spectral features of clouds and smoke. Although the SLOPE1 index is not used explicitly in the classification of pixels values for smoke detection it was found to provide useful information post-classification in smoke-plume characteristics, particularly in identification of the near-source (fire) end of the plume.

In examining the smoke masks a number of false detections (pixel values > 0) were observed in association with shallow water features and water bodies with high total suspended matter content. These comprised largely beach zones and turbid estuaries such as the Coorong, the Fitzroy, some rivers, etc. To reduce the number of false positives of smoke, cloud and other aerosols, and to improve training dataset integrity, state land boundaries and surface hydrology polygons were used to produce a vector mask removing remove ocean and land water features from aerosol classification band by setting these pixel values to zero.

The procedure for producing the "land surface mask" was:

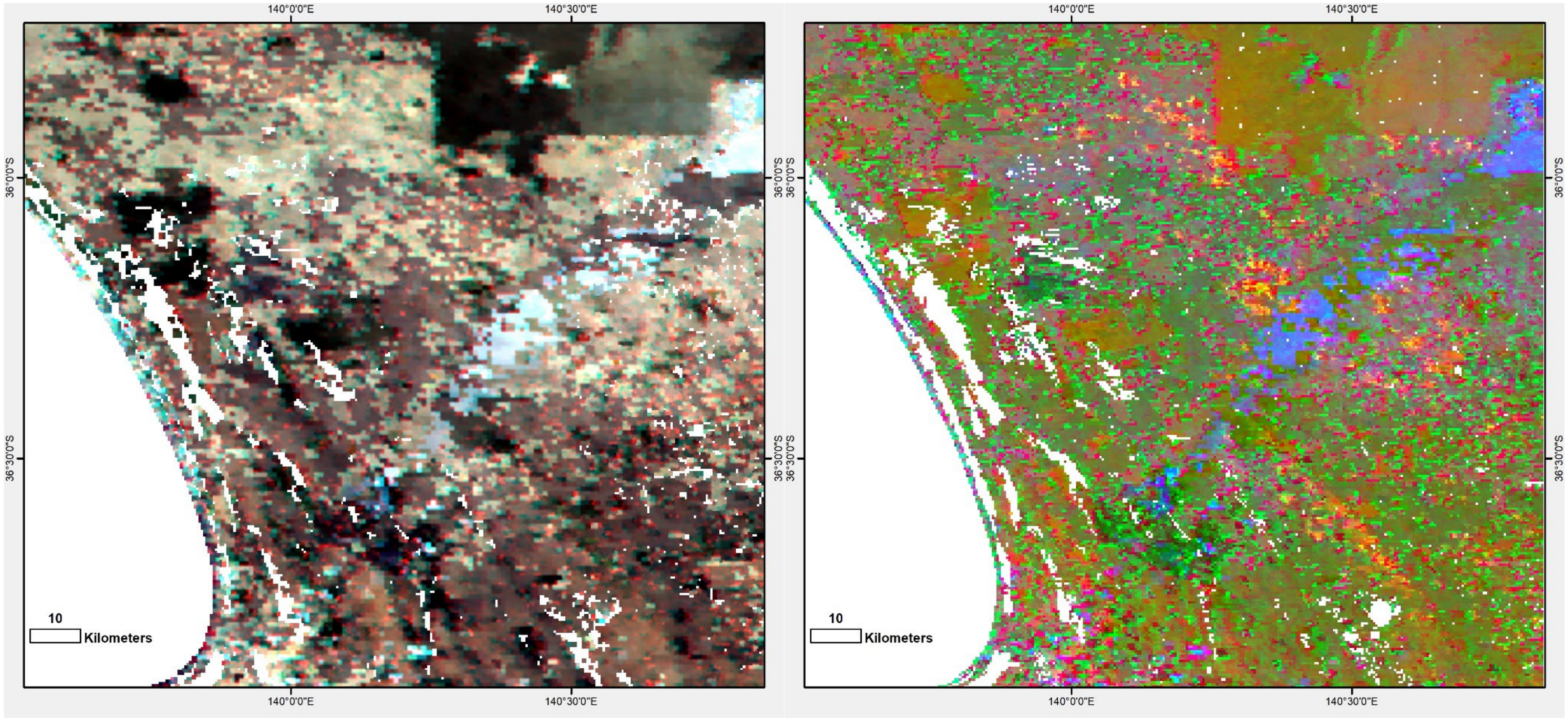
- Utilise vector shapefile of state boundary from local state government data platform.
- Buffer state boundary by -200 m to remove 200 m of land-beach zone.
- Utilize the National Surface Hydrology Polygon (Regional) dataset[9], published 2015, to clip out surface water features from the buffered stated boundary. This represents the final "land surface mask" where all remaining areas in the mask correspond to land surface features and can be retained in the model.
-

This step will be less necessary when real HyperScout-2 spectral data becomes available, as the acquisition of multiple bands around 0.97 μm and 1.2 μm will capture strong water absorptions features at these wavelengths [e.g. [32]] and therefore provide greater sensitivity to discriminating surface (and atmospheric) water features.

An example of a pseudo-HyperScout-2 image with a "HyperScout-2 Smoke Detection Algorithm" generated smoke mask is shown in Figure 2-4. The HS2 simulated image is over a fire event near the Coorong in South Australia on December 31st, 2020. The three image panels shown display:

- a 'natural colour' composite in red-green-blue wavelengths (left panel). Here the smoke plume appears in blue-ish grey tones near source, and white higher in the plume (visibly similar to cloud in other parts of the image). Burn scars are also hard to visibly separate from dark vegetation.
- a false colour composite (right panel), designed to highlight smoke and cloud and distinguish them from the background utilising the SLOPE1 and SLOPE2 indices. Here the smoke plume is dark indigo near source and purple further up the plume, while clouds in the remainder of the image are magenta. Photosynthetically active vegetation is bright orange, while burn scars are dark teal.
- The HSSDA mask over the 'natural colour' composite (bottom panel). Here the two classes containing smoke are shown in red and orange. The plume is clearly detected, with the densest smoke near source classified as "smoke endmember". Smoke is not confused as being cloud. False detections occur near the shoreline.

[9] https://ecat.ga.gov.au/geonetwork/srv/eng/catalog.search#/metadata/83134



*see caption on next page

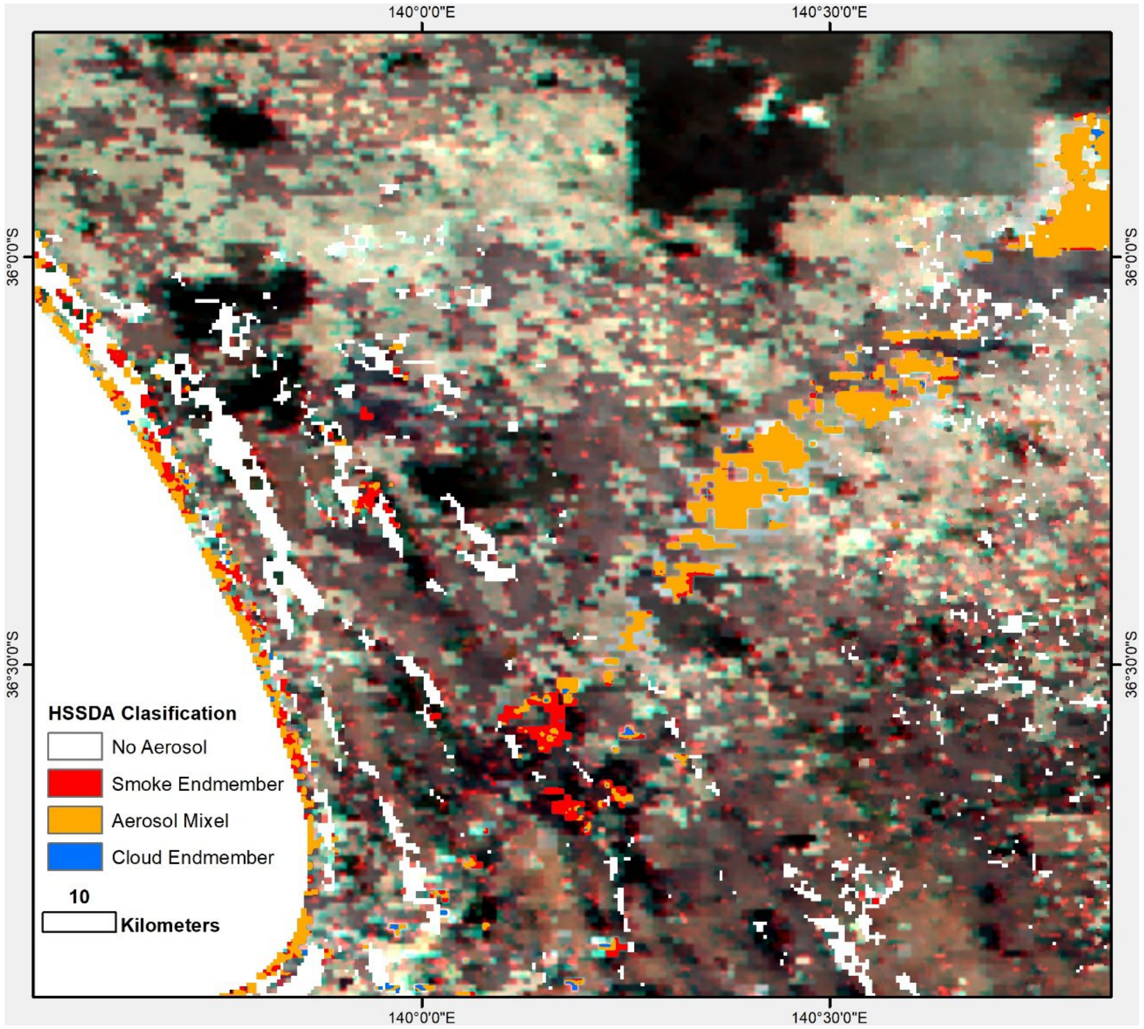


FIGURE 2-4: EXEMPLARY SIMULATED HYPERSCOUT-2 IMAGERY AND SMOKE DETECTION OVER A FIRE EVENT NEAR COORONG IN SOUTH AUSTRALIA ON DECEMBER 31ST 2020. PANEL 1: 'NATURAL COLOUR' IMAGE; PANEL 2: 'FALSE COLOUR' USING SPECTRAL SLOPE1 AND SLOPE2 INDICES DESCRIBED IN TEXT TO HIGHLIGHT SMOKE AND CLOUD; PANEL 2: HSSDA AUTOMATED SMOKE DETECTION.

2.4. Producing AI Fire Smoke Training Dataset

A final remote sensing goal of this project was the production of a tiled and labelled training dataset at HyperScout-2 spectral and spatial resolution, for the purpose of training an AI-based fire smoke detection model. There are currently many labelled fire smoke datasets consisting of labelled ground camera (e.g. handheld or surveillance) or aerial RGB photos (e.g. the Corsica dataset [33], the Kaggle "Fire and Smoke Dataset"[10]). There are also a range of existing multispectral labelled satellite datasets, including the "USTC_SmokeRS" at MODIS spatial resolution of 1 km [6] and "FASDD: Flame And Smoke Detection Dataset" which includes amongst other image sources Landsat-8 and Sentinel-2 satellite imagery at 10-30m/pixel for Deep Learning in Fire Detection [34]. However, there are no smoke labelled datasets at hyperspectral resolution appropriate for training an AI model to be applied to the HS2 sensor.

The previous sections have outlined the production of simulated HyperScout-2 imagery, and of the "HyperScout-2 Smoke Detection Algorithm" (HSSDA) that classifies pixels into classes of "smoke", "cloud" and "aerosol mixels". In order to develop a training dataset for the neural network smoke detection model, several key principals were followed:

- Aim to distinguish 4 classes as described above: smoke, cloud, mixtures of aerosols, and with the remaining pixels being assigned to background LULC.
- Aim to capture as much variance within each of those 4 classes so that the model is robust in a real-life scenario.
- Encompass as much non-class related variance in the image data related to the HS2 satellite model, so that the model is also robust to the variations in changes to image acquisition (e.g. scale, lighting changes, season, contrast, image noise).

To achieve these aims 6 regions of interest were chosen across 5 states and territories, as shown in Figure 2-5, encompassing over 53 million hectares of land surface area. Image dates were chosen to overlap historical fire events described in Table 4, resulting in the processing of over 500 VIIRS satellite images over 200 dates, and bushfire events over 3 years from 2018-2020. This broad range in spatial and temporal characteristics in the training data was chosen as it allows the capture of different seasonal bushfire events, occurring in different climactic zones, with imagery captures also at different times of day, and viewing angles. This heterogeneity is crucial for the robustness of an AI smoke detection model.

[10] https://www.kaggle.com/datasets/dataclusterlabs/fire-and-smoke-dataset

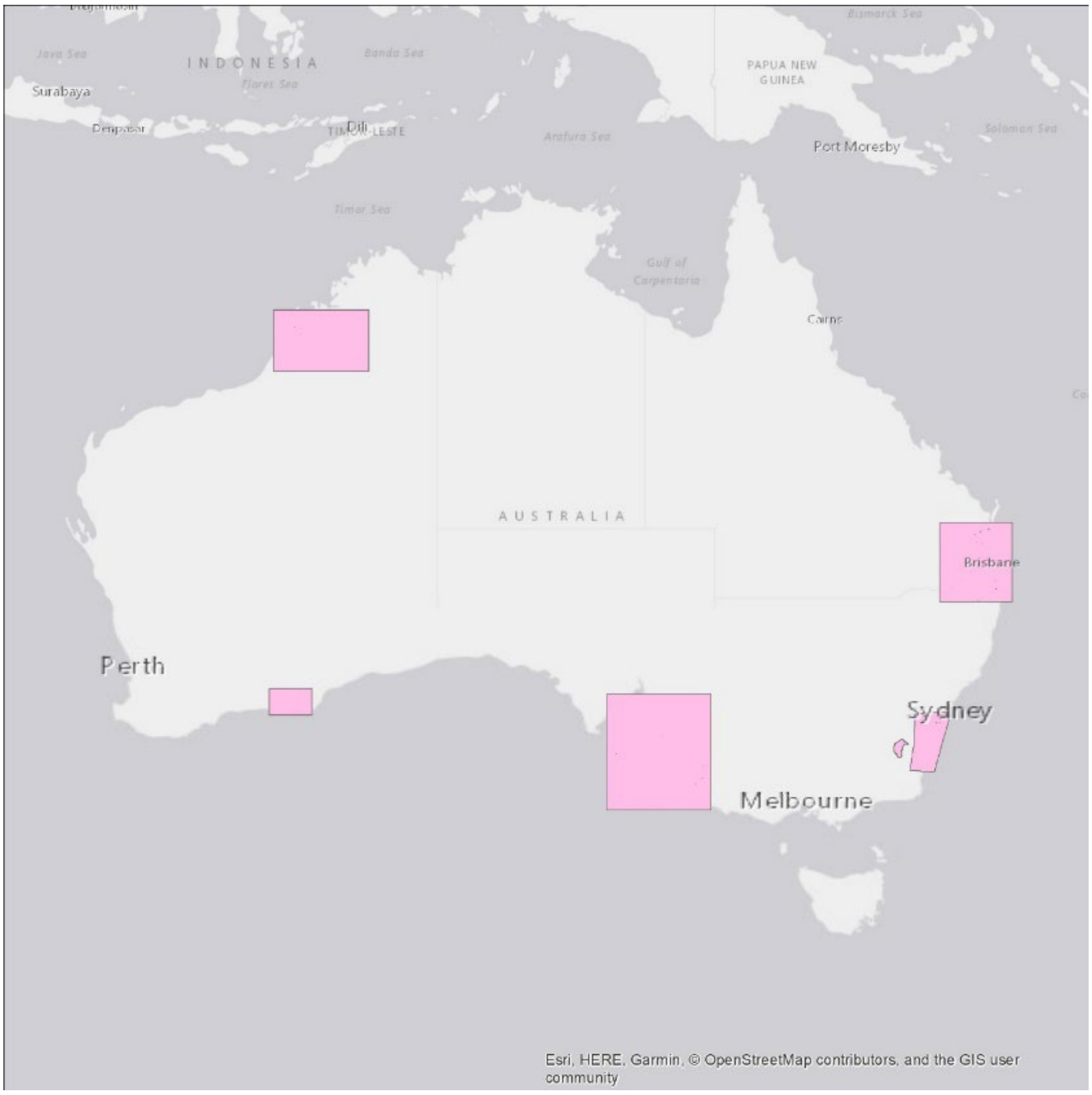


Figure 2-5: Locations of simulated HyperScout-2 imagery: Queensland, New South Wales, ACT, South Australia, and two regions in Western Australia (southern: Esperance; northern: Broome).

TABLE 4: IMAGE AND TILE NUMBERS FOR REGIONS OF INTEREST.

Site	Bushfire Event	Timespan	Days	Number of tiles	Total storage (Gb)[11]
ACT	Started 16th January 2019. Extinguished 27th February. https://knowledge.aidr.org.au/resources/black-summer-bushfires-act-2020/	20191201-20200229	71	2616	4.1
NSW	Ongoing throughout December 2019-February. https://knowledge.aidr.org.au/resources/black-summer-bushfires-nsw-2019-20/	20191201-20200228	70	22728	285.9
SA	Started 24th October 2019. Fire conditions resolved in January. https://knowledge.aidr.org.au/resources/black-summer-bushfires-sa-2019-20/	20191020-20200201	102	84235	1059.5
QLD	Started 8th October 2019. Fire conditions resolved in January. https://knowledge.aidr.org.au/resources/black-summer-bushfires-qld-2019/	20191020-20191231	72	48964	615.9
WA-ESP	Started prior to Feb. 22nd 2019. Extinguished March. https://www.abc.net.au/news/2019-02-28/catastrophic-fire-warning-ahead-of-wa-long-weekend/10859008 https://knowledge.aidr.org.au/resources/2019-bushfire-wa-esperance-complex-bushfires/	20190202-20190305	32	5008	63.0
WA-BRM	Started October 11th 2018. Extinguished October 18th. https://www.abc.net.au/news/2018-10-20/massive-area-in-kimberley-torched-by-out-of-control-bushfire/10398450	20181011-20181125	48	26413	332.2
Total				189964	

Tiles were produced by partitioning the simulated HS2 imagery, at 75 m/pixel resolution, 48 spectral bands, and 1 additional band consisting of the HSSDA mask, into square grids containing 256x256 pixels (in the x-y dimensions). Each tile covers a region of 19.2 x 19.2 km (36864 ha) on the ground. The square grids, or tiles, were arranged so that they had 50% overlap in both the x and y direction, as shown in Figure 2-6. Tiles were only kept for model training if they did not contain any of the image null pixel value border, in other words all 256x256 pixels in each dimension must contain valid spectral information.

[11] Note that the storage size of a single HSSDA smoke classification tile is 257 kb.

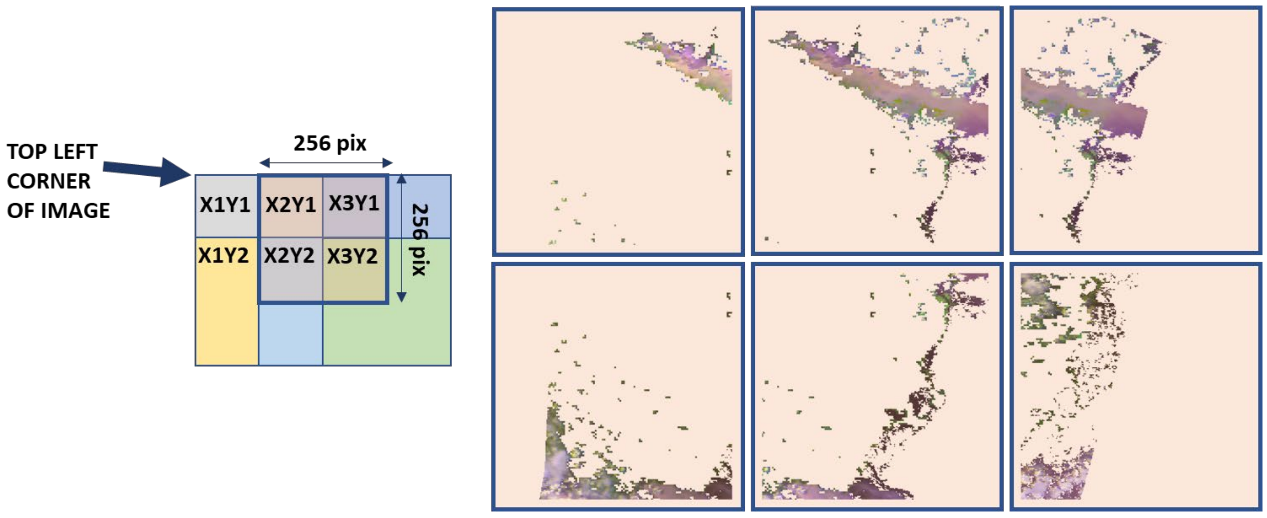
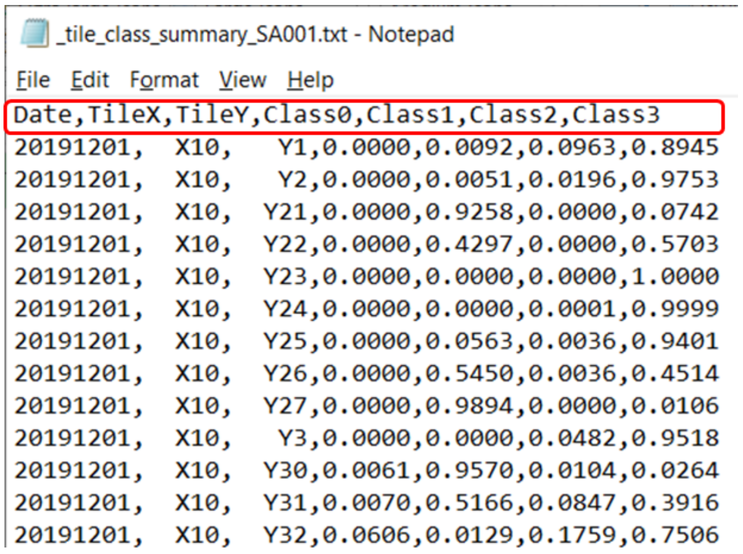


FIGURE 2-6: SCHEMATIC SHOWING TILE PRODUCTION. THE FIGURE ON THE LEFT SHOWS A NAVY BLUE MOVING WINDOW THAT CUTS THE 256X256 PIXEL TILES OUT OF THE SIMULATED HS2 IMAGE, WITH EACH TILE SHARING A 50% OVERLAP WITH AT LEAST ONE OTHER TILE IN BOTH THE X AND Y DIMENSIONS. THE SIX COLOURED TILES SHOWN ON THE LEFT (X1Y1,X2Y1…..X2Y2) ARE VISUALIZED ON THE RIGHT SHOWING THE DETECTED SMOKE FEATURES.

To ensure that an even distribution of tiles is used for each training class for the model, a text file was also produced giving the statistics of class coverage within each tile. An example is shown in Figure 2-7. This provides the ability to select tiles by class coverage, ensuring that the model sees a statistically representative sample of smoke (class 1), cloud (class 3), background land-cover (class 0), and mixtures of aerosols (class 2). Without this the model would easily become biased in training. The histogram of all tiles from a single ROI is shown in Figure 2-8, for the ACT region. It reveals that for the vast majority of tiles in this area, smoke covers only a small spatial fraction of the tile, typically less than 5% of the tile area, and very rarely more than 20%. On the other hand, a much more significant number of tiles have a higher cloud coverage, with over 15% of all tiles having greater than 50% of their tile area classified as cloud. If these statistics were not considered and tiles were randomly sampled for training, then the machine learning model would become easily biased to cloud detection rather than smoke.

The final HSSDA dataset comprises 189,964 labelled hyperspectral tiles with classes differentiating “smoke”, “cloud”, and “mixtures of smoke, haze, cloud and other aerosols” for the specific purpose of training an AI smoke detection model for hyperspectral data.



_tile_class_summary_SA001.txt - Notepad
File Edit Format View Help

```
Date,TileX,TileY,Class0,Class1,Class2,Class3
20191201,   X10,    Y1,0.0000,0.0092,0.0963,0.8945
20191201,   X10,    Y2,0.0000,0.0051,0.0196,0.9753
20191201,   X10,   Y21,0.0000,0.9258,0.0000,0.0742
20191201,   X10,   Y22,0.0000,0.4297,0.0000,0.5703
20191201,   X10,   Y23,0.0000,0.0000,0.0000,1.0000
20191201,   X10,   Y24,0.0000,0.0000,0.0001,0.9999
20191201,   X10,   Y25,0.0000,0.0563,0.0036,0.9401
20191201,   X10,   Y26,0.0000,0.5450,0.0036,0.4514
20191201,   X10,   Y27,0.0000,0.9894,0.0000,0.0106
20191201,   X10,    Y3,0.0000,0.0000,0.0482,0.9518
20191201,   X10,   Y30,0.0061,0.9570,0.0104,0.0264
20191201,   X10,   Y31,0.0070,0.5166,0.0847,0.3916
20191201,   X10,   Y32,0.0606,0.0129,0.1759,0.7506
```

Figure 2-7: Text file giving the fraction of class coverage in each spatial tile, where class 1 = "smoke endmember", and class 3 = "cloud endmember".

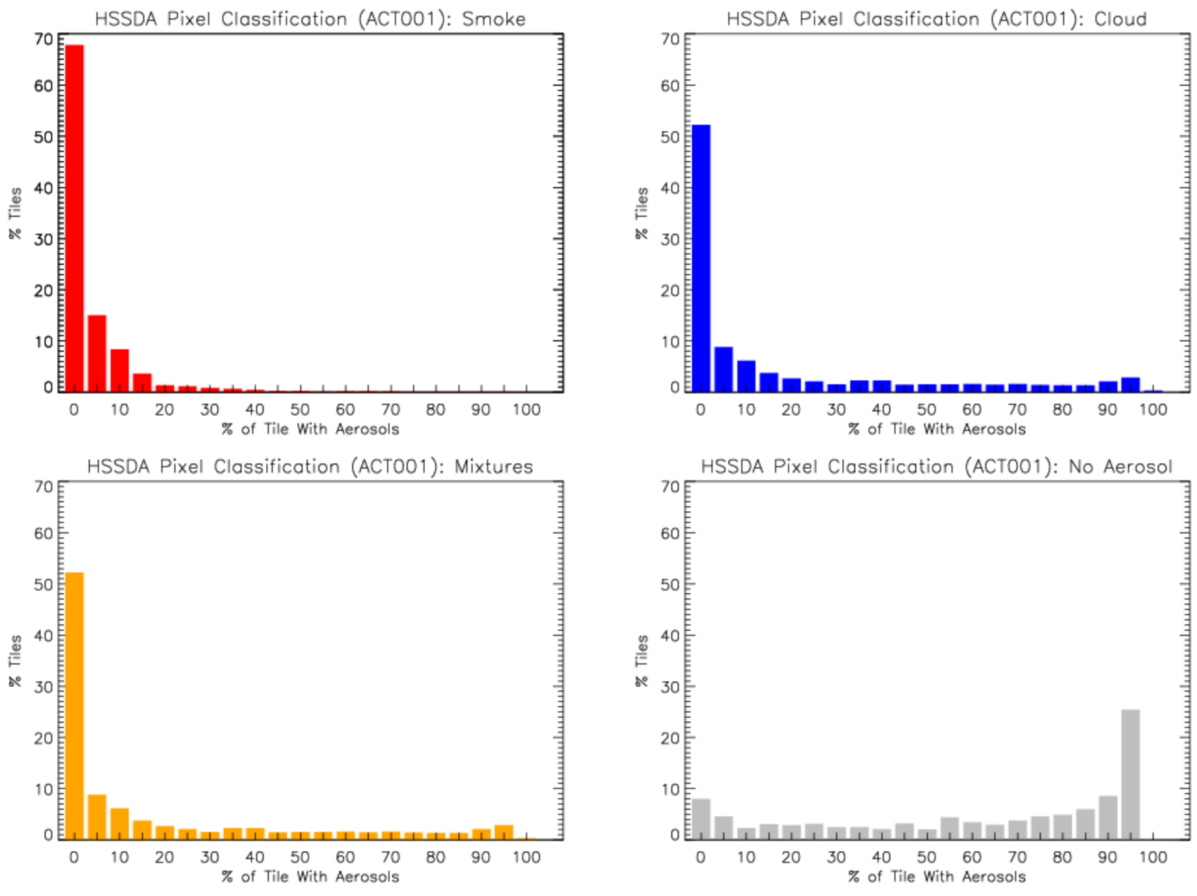


Figure 2-8: Percentage of smoke coverage in all tiles for the Australian Capital Territory imagery during the Black Summer fires of 2019. All horizontal and vertical axes are on the same scale.

2.5. Manual Fire Smoke Labelling

For comparison to the automatically generated HSSDA smoke detections, and the AI based smoke detection model, a number of HS2 simulated images were manually labelled, through a process of visual inspection, for “smoke”, “cloud”, and “smoke cloud mixtures”, see Table 5 and Figure 2-9. Areas identified as belonging to each of the three classes were manual outlined with their boundary stored in a polygon shapefile. Cloud was generally identified through its visual brightness and form/shape, smoke through its plume shape, origination from a clear source, and blueish and brownish tones. Smoke cloud mixtures were outlined where the two features appeared to overlap or were hard to distinguish. The vector polygons were then converted to rasters with the same co-ordinate reference system and spatial pixel size as the HSSDA dataset, for direct comparison. This human labelling dataset provides an independent comparison to the output of both the HSSDA and machine learning models.

TABLE 5: MANUAL LABELLING OF SMOKE AND CLOUD.

Site	Class	# labelled pixels
ACT	Class 1 ⇨ “smoke endmembers” Class 2 ⇨ “aerosol mixels” Class 3 ⇨ “cloud endmembers”	100983 9720 62048
NSW	Class 1 ⇨ “smoke endmembers” Class 2 ⇨ “aerosol mixels” Class 3 ⇨ “cloud endmembers”	480318 180328 272816
SA	Class 1 ⇨ “smoke endmembers” Class 2 ⇨ “aerosol mixels” Class 3 ⇨ “cloud endmembers”	193362 373435 347114
QLD	Class 1 ⇨ “smoke endmembers” Class 2 ⇨ “aerosol mixels” Class 3 ⇨ “cloud endmembers”	264796 745402 272679
WA-ESP	Class 1 ⇨ “smoke endmembers” Class 2 ⇨ “aerosol mixels” Class 3 ⇨ “cloud endmembers”	38467 22507 42386
WA-BRM	Class 1 ⇨ “smoke endmembers” Class 2 ⇨ “aerosol mixels” Class 3 ⇨ “cloud endmembers”	239821 56138 196679

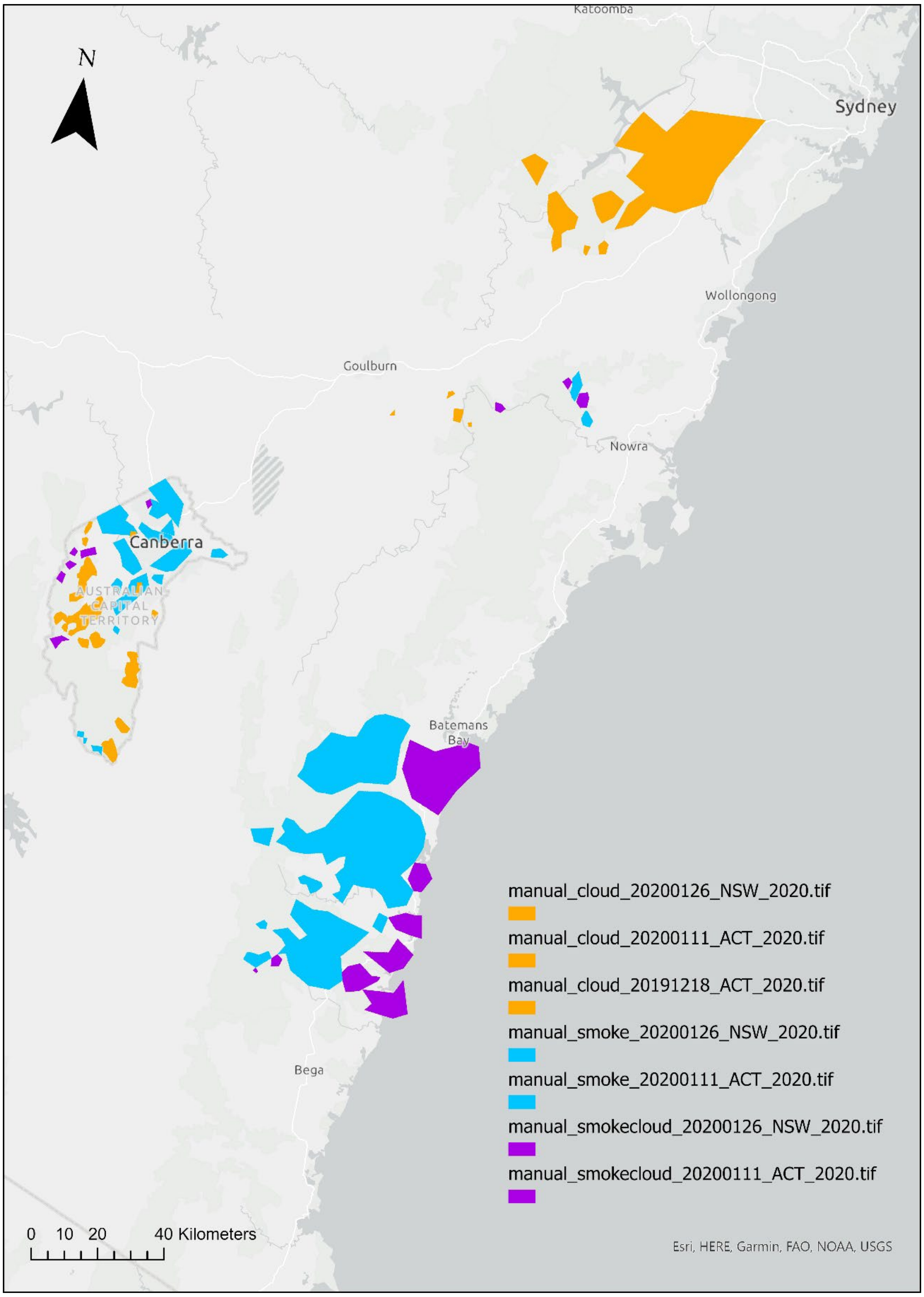

Katoomba
N
Sydney
Wollongong
Goulburn
Nowra
Canberra
AUSTRALIAN CAPITAL TERRITORY
Batemans Bay
Bega
manual_cloud_20200126_NSW_2020.tif
manual_cloud_20200111_ACT_2020.tif
manual_cloud_20191218_ACT_2020.tif
manual_smoke_20200126_NSW_2020.tif
manual_smoke_20200111_ACT_2020.tif
manual_smokecloud_20200126_NSW_2020.tif
manual_smokecloud_20200111_ACT_2020.tif
0 10 20 40 Kilometers
Esri, HERE, Garmin, FAO, NOAA, USGS

FIGURE 2-9: EXEMPLARY MANUALLY LABELLED CLOUD/SMOKE TRAINING DATASET.

3. Onboard Processing

To evaluate the performance of onboard processes, we build an emulation system to simulate the computing environment of the Kanyini satellite. For the hardware, the emulation system uses a Raspberry Pi and an Intel Neural Compute Stick (NCS). We align the emulation system through software configurations to closely match the Kanyini satellite specifications. In terms of software, we have developed a pipeline that facilitates performance measurement, including running time, memory usage, and power consumption, for a wide range of onboard tasks.

3.1. Emulation system architecture: Motivation and Aims

In the early stages of satellite missions, the computing environment is often not yet available. However, it is important to evaluate the capabilities and resource requirements of onboarding processes and algorithms early on for research projects. To deal with the problem, we have developed an emulation system that simulates onboard processes and evaluates their performance.

The emulation system is built using a Raspberry Pi and an Intel NCS for its hardware components. We align the emulation system through software configurations to closely match the satellite specifications. In terms of software, we have developed a pipeline that facilitates performance measurement, including runtime, memory usage, and power consumption, for a wide range of onboard tasks. It is worth noting that although the emulation system is configured to match the Kanyini satellite, it is designed to be adaptable and easily applicable to other missions.

3.2. Hardware

A satellite computing environment typically consists of three levels: spacecraft, sensor, and AI module. In the Kanyini satellite, the sensor employed is HyperScout2, and the AI module utilizes the Eyes of Things (EOT) board developed by Ubotica. The EOT board is equipped with Intel`s Movidius Myriad 2 vision processing units (VPUs). Figure 3-1 provides a visual representation of these components.

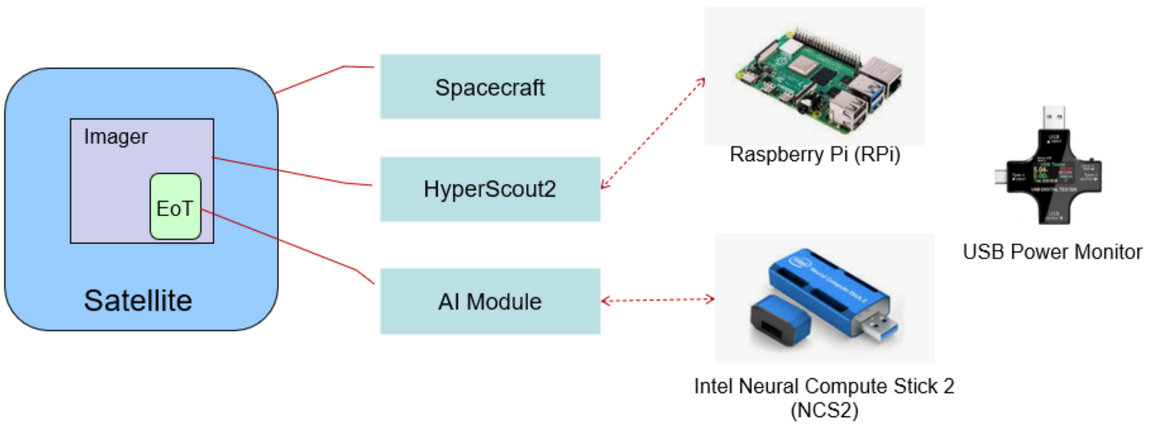


FIGURE 3-1. THE HARDWARE OF THE EMULATION SYSTEM.

The emulation system focuses on the sensor and the AI module. We chose hardware devices whose specifications are close to or slightly above those of the satellite environment, then used software configuration to tune settings. As shown in Figure 4-1, a Raspberry Pi (RPi), a single-board computer, is used to emulate a HyperScout2 sensor and an Intel Neural Compute Stick 2 (NCS2) to simulate the AI module. The major differences between the Kanyini satellite and the selected devices are presented in Appendix B.2 (confidential). The CPU frequency, the number of CPU cores, RAM, and the number of VPU cores of the emulation system are adjusted and aligned with the Kanyini environment by software configurations during the emulation experiments. The corresponding codes are provided in Appendix B.3 (confidential).

To measure the power consumption a USB power monitor is used to measure the power consumption of onboard processing, which is connected between the power supply and the Raspberry Pi.

3.3. Software

3.3.1. The Emulation Pipeline

As shown in Figure 3-2, the emulation pipeline includes four modules: (1) performance measuring; (2) resource constraint; (3) onboard processing; and (4) results analysis and model optimization. The four modules form a pipeline to conduct performance testing on various onboard processing.

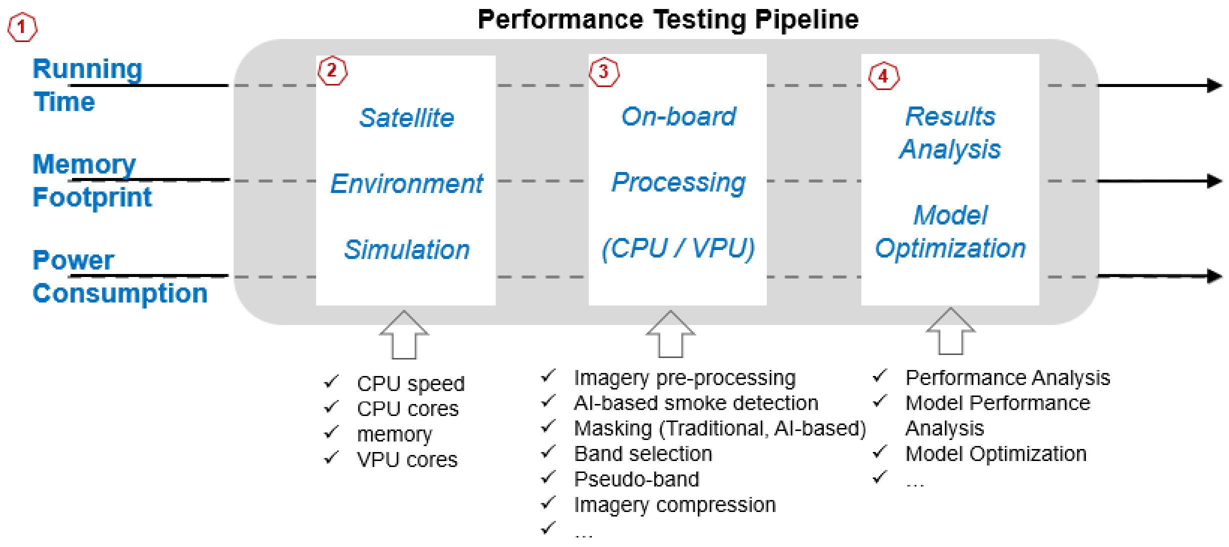


FIGURE 3-2. THE EMULATION PIPELINE

The performance measuring module provides a framework for measuring the running time, memory footprint and power consumption of various onboard processes. The implemented code is emulate.py. It is noted that the power consumption measurement is currently done manually, as the USB power monitor does not have a software interface. It is possible to upgrade the hardware and measure power consumption via software in the future research phase.

The resource constraint module applies the constraints on the hardware of the emulation system to make it match the Kanyini environment as closely as possible. The resource includes the CPU frequency of RPi, the number of CPU cores of RPi, the RAM of PRI and the number of VPU cores (SHAVES) of NCS2.

The onboard processing module implements various onboard processes, including non-AI and AI tasks. This module also contains some code to support the onboard processing. This module accounts for the largest percentage of the whole software package and is still increasing along with the development of onboard processing. The details of the code of the module are presented in the Onboard Processing Tasks section.

The results analysis and model optimization module present and analyses the testing results, producing the AI model optimization recommendations. As there are not enough testing results at this stage, the code in the module is still under construction.

A bash script file, *test_batch.sh*, starts an emulation experiment. The resource constraint is first conducted, followed by the onboard process procedure of (1) tiling an image, (2) making AI inference, (3) generating a mask according to the results of AI inference, (4) merging tiles with detected smoke, (5) compressing the merged image, and (6) splitting the compressed image into small files of the size of 512 KB, waiting for downlinking. The constrained resources are recovered at the last. Exemplary script of an emulation experiment is provided in Appendix B.4 (confidential).

3.3.2. Lightweight AI for onboard fire smoke detection

The VIB_SD model [35] is a lightweight model designed specifically for fire smoke detection using multi-spectral satellite imagery. VIB_SD integrates two key modules, respectively the Inception-Attention Module and the Inception-Residual Module, to facilitate residual learning and feature extraction at multiple scales, allowing an accurate detection of fire smoke in variant scopes captured by in the satellite imagery, even if the fire smoke is mixed with other types of aerosols such as cloud or dust. We chose VIB_SD as the onboard AI model for our emulation experiments for two primary reasons. First, VIB_SD is resource-efficient, featuring approximately 1.6 million parameters, in contrast to other state-of-the-art models that can have over 50 million parameters. This lightweight architecture makes it well-suited for onboard satellite detection. Second, VIB_SD has demonstrated high prediction accuracy in fire smoke detection tasks using multi-spectral satellite imagery.

Readers can refer to Zhao, Liu [35] for more details about the VIB_SD model.

> *Zhao, Liang, Jixue Liu, Stefan Peters, Jiuyong Li, Simon Oliver, and Norman Mueller. 2022. "Investigating the Impact of Using IR Bands on Early Fire Smoke Detection from Landsat Imagery with a Lightweight CNN Model." Remote Sensing 14 (13). doi: 10.3390/rs14133047.*

3.3.3. AI Model Preparation

The AI onboard involves the use of the OpenVINO toolkit. The OpenVINO (Open Visual Inference and Neural network Optimization) toolkit is an open-source toolkit that supports optimizing a deep learning model from various frameworks and deploys it on a range of Intel processors and other hardware platforms, including NCS2 in our emulation system. See also:

https://docs.openvino.ai/2023.0/get_started.html

Figure 3-3 shows that the model preparation for AI onboard involves three steps: model training, model converting and model inference. The first two steps are conducted on-ground, while the model inference is performed onboard. In the model training step, a model is trained and saved on a host computer. Then in the model converting step, the trained model is transformed into a graph representation of OpenVINO, named Intermediate Representation, using OpenVINO optimizer. The IR formats of a model contain two files, an XML file (.xml) to represent the topology of a network and a binary file (.bin) for the parameters. The two files are deployed to NCS2 to make model inferences.

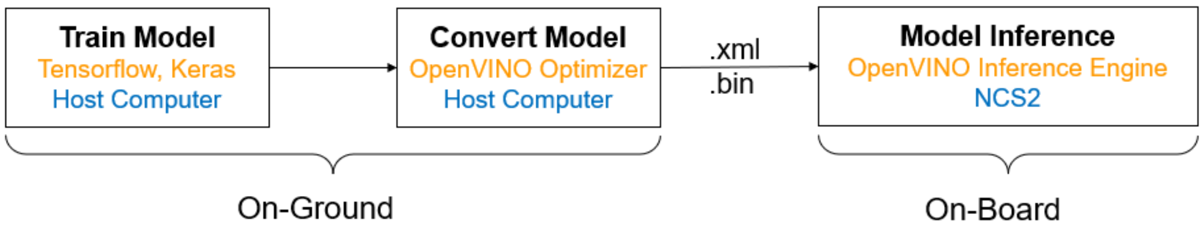


FIGURE 3-3. MODEL PREPARATION FOR AI ONBOARD

The commands to convert a trained model on a Windows host computer are presented in Appendix B.5 (confidential).

3.3.4. Onboard Processes

The emulation system has implemented the following onboard processes.

- Compress imagery: code in file *compress_split.py*. The emulation system supports the following compression methods: LZMA, LZW, PackBits, Deflate, PIXTIFF, LERC, Zstd, JPEG, JPEG2000, JPEGXL, and PNG. The Kanyini satellite uses LZMA to compress imagery, so we set it as the default compression method. We also conduct experiments to compare the compression ratio and efficiency of these compression methods. The comparison results are presented in Section 3.4.
- Split imagery into small fills of size 512 KB: code in file compress_split.py.
- Partition imagery: code in file tiling.py. The default tile dimension is 256 by 256.
- Extract specified bands from imagery: code in file extract_bands.py.
- Merge tiles into imagery: code in file tiling.py.
- Onboard inference using VIB_SD model: code in file inference.py.

3.4. Experiments and Results

3.4.1. Comparison of Compression Methods

The emulation system supports 11 compression methods. We summarize the main features of these methods as the following.

Here's a comparison of some popular compression methods:

1. LZMA:
   - High compression ratio.
   - Slower compression and decompression speed.
   - Suitable for compressing large files or archives.
2. LZW:
   - Used in GIF image format.
   - Good compression ratio for indexed color images.
   - Simple and fast compression and decompression.
   - Patent-encumbered, limiting its usage in some contexts.
3. PackBits:
   - Simple compression method used in TIFF image format.
   - Provides modest compression ratios.
   - Fast compression and decompression speed.

Limited effectiveness for certain types of data.

4. Deflate:
   - Widely used in formats like ZIP, gzip, and PNG.
   - Good compression ratio and fast compression/decompression speed.
   - Offers a balance between compression effectiveness and speed.
   - Provides adjustable compression levels.
5. PIXTIFF:
   - A proprietary compression method used in GeoTIFF files.
   - Optimized for aerial and satellite imagery.
   - Can provide high compression ratios for large images.
   - Limited support and compatibility compared to widely-used formats.
6. LERC:
   - Used for lossless compression of raster data in Esri's ArcGIS.
   - Designed for efficient compression of elevation and imagery data.
   - Provides high compression ratios while preserving data fidelity.
   - Suitable for large geospatial datasets.
7. Zstd:
   - Developed as a replacement for Deflate with better compression ratios.
   - Fast compression and decompression speed.
   - Offers adjustable compression levels and tunable trade-offs.
   - Suitable for a wide range of data types and sizes.

8. JPEG:
   - Lossy compression method for images.
   - Provides high compression ratios but introduces image artifacts.
   - Suitable for continuous-tone images with complex content.
   - Popular for web images and digital photography.
9. JPEG2000:
   - Advanced image compression standard.
   - Supports both lossless and lossy compression.
   - Provides excellent compression efficiency with various image types.
   - Offers progressive transmission and region of interest coding.
10. JPEGXL:
    - Next-generation image compression format.
    - Supports both lossless and lossy compression.
    - Promises superior compression efficiency compared to previous standards.
    - Introduces features like animation, transparency, and lossless transcoding.
11. PNG:
    - Lossless compression method for images.
    - Provides good compression ratios for images with large areas of solid color.
    - Uses Deflate compression algorithm.
    - Supports transparency and lossless image data.

To compare the performance of the 11 compression methods, we conduct experiments using two synthetic images with dimensions matching those of Kanyini and present the file size after compression and the running time.

The first input image has dimensions of 2560 by 1856, consisting of 45 bands, and has a file size of 407.82 MB. The compression results of the 11 methods are the following.

1.	LZMA:	File size = 323.82 MB,	Time = 62.30 seconds
2.	LZW:	File size = 476.11 MB,	Time = 1.48 seconds
3.	PackBits:	File size = 411.06 MB,	Time = 1.73 seconds
4.	Deflate:	File size = 349.82 MB,	Time = 3.20 seconds
5.	PIXTIFF:	File size = 349.82 MB,	Time = 3.58 seconds
6.	LERC:	File size = 325.3 MB,	Time = 6.30 seconds
7.	Zstd:	File size = 349.0 MB,	Time = 2.39 seconds
8.	JPEG:	File size = 285.48 MB,	Time = 9.78 seconds
9.	JPEG2000:	File size = 335.48 MB,	Time = 25.13 seconds
10.	JPEGXL:	File size = 324.63 MB,	Time = 15.26 seconds
11.	PNG:	File size = 362.6 MB,	Time = 9.29 seconds

The second input image has dimensions of 3072 by 1856, consisting of 45 bands, and has a file size of 522.01 MB. The compression results of the 11 methods are the following.

1. LZMA: File size = 414.48 MB, Time = 87.63 seconds
2. LZW: File size = 609.41 MB, Time = 2.81 seconds
3. PackBits: File size = 526.15 MB, Time = 3.04 seconds
4. Deflate: File size = 447.76 MB, Time = 4.19 seconds
5. PIXTIFF: File size = 447.76 MB, Time = 4.24 seconds
6. LERC: File size = 416.36 MB, Time = 8.29 seconds
7. Zstd: File size = 446.72 MB, Time = 3.15 seconds
8. JPEG: File size = 365.09 MB, Time = 12.19 seconds
9. JPEG2000: File size = 429.41 MB, Time = 37.84 seconds
10. JPEGXL: File size = 415.53 MB, Time = 21.54 seconds
11. PNG: File size = 464.11 MB, Time = 12.72 seconds

The experiments were conducted on a Windows 10 computer with an Intel i7 2.8 GHz processor and 16 GB of RAM. The experimental results demonstrate that LZMA achieves the highest compression ratio but significantly longer running times than other methods.

Emulation experiments were conducted using various parameters, including different numbers of pixels across the track for visual and near-infrared bands (selected from 3072 and 2560), different numbers of bands (selected from 45 and 48) and different numbers of tiles (selected from 5, 10, 20 and 40). The results of different onboard processes with these different configurations are presented in Table 6, including the number of pixels of VNIR bands on across the track (#ACT); the input file size in MB (Input); the output file size in MB (Output); the number of bands (#Bands); the number of tiles (#Tiles); running time in seconds (Runtime); the average memory in MB (Mem-A); the peak memory in MB (Mem-P); and the power consumption in Watt (Power).

TABLE 6. ONBOARD PROCESSING PERFORMANCE RESULTS FOR DIFFERENT PROCESSES AND CONFIGURATIONS

Task	#ACT	Input (MB)	Output (MB)	#Bands	#Tiles	Runtime (sec)	Mem-A (MB)	Mem-P (MB)	Power (Watt)
compress	3072	466.72	36.38	45	5	462.96	22.63	505.46	0.56
compress	3072	466.72	57.73	45	10	655.72	21.89	505.05	0.44
compress	3072	466.72	100.25	45	20	691.35	22.20	505.20	0.42
compress	3072	466.72	185.31	45	40	739.54	24.62	507.56	0.52
compress	2560	388.94	33.88	45	5	538.53	22.04	427.08	0.48
compress	2560	388.94	55.18	45	10	551.94	22.49	427.54	0.52
compress	2560	388.94	97.79	45	20	575.37	22.31	427.35	0.48
compress	2560	388.94	140.06	45	30	602.16	23.14	428.24	0.49
compress	2560	388.94	182.44	45	40	562.19	25.17	430.36	0.45
compress	3072	466.72	370.59	45	45	833.86	21.11	504.56	0.47
compress	2560	388.94	308.83	45	45	690.65	21.04	426.82	0.46
compress	3072	466.72	142.67	45	30	527.43	22.26	506.94	0.51
merge_tiles	3072	466.72	466.72	45	5	6.29	28.27	495.36	0.53
merge_tiles	3072	466.72	466.72	45	10	6.18	24.83	495.20	0.49
merge_tiles	3072	466.72	466.72	45	20	6.24	25.33	494.95	0.44
merge_tiles	3072	466.72	466.72	45	30	6.16	27.88	494.96	0.46
merge_tiles	3072	466.72	466.72	45	40	6.11	28.01	495.09	0.45

merge_tiles	2560	388.94	388.94	45	5	5.29	27.93	417.15	0.47
merge_tiles	2560	388.94	388.94	45	10	5.27	28.15	417.46	0.47
merge_tiles	2560	388.94	388.94	45	20	5.20	28.00	417.30	0.43
merge_tiles	2560	388.94	388.94	45	30	5.14	27.94	417.24	0. 44
Task	**#ACT**	**Input (MB)**	**Output (MB)**	**#Bands**	**#Tiles**	**Runtime (sec)**	**Mem-A (MB)**	**Mem-P (MB)**	**Power (Watt)**
merge_tiles	2560	388.94	388.94	45	40	5.03	28.12	417.43	0.41
smoke_mask	3072	466.72	10.38	45	10	2.73	27.84	502.65	0.51
smoke_mask	2560	388.94	8.64	45	10	2.35	27.63	423.59	0.48
split	3072	36.38	0.48	45	5	0.68	21.46	58.24	0.44
split	3072	57.73	0.48	45	10	1.09	21.36	79.43	0.52
split	3072	100.25	0.48	45	20	1.89	21.51	122.11	0.48
split	3072	185.31	0.48	45	40	3.51	21.55	207.23	0.42
split	2560	33.88	0.48	45	5	0.66	21.41	55.64	0.45
split	2560	55.18	0.48	45	10	1.06	21.73	77.34	0.41
split	2560	97.79	0.48	45	20	1.84	21.51	119.66	0.42
split	2560	140.06	0.48	45	30	2.62	21.64	162.07	0.5
split	2560	182.44	0.48	45	40	3.45	21.50	204.22	0.46
split	3072	370.59	0.48	45	45	6.71	21.20	392.33	0.52
split	2560	308.83	0.48	45	45	5.69	21.64	331.01	0.48
split	3072	142.67	0.48	45	30	2.58	21.43	164.63	0.46

tiling&extract_bands	3072	466.72	0.36	3	84	3.48	31.77	499.02	0.47
tiling&extract_bands	3072	466.72	0.49	4	84	3.63	31.37	499.59	0.44
tiling&extract_bands	3072	466.72	0.60	5	84	3.64	31.22	499.69	0.46
tiling&extract_bands	2560	388.94	0.36	3	70	2.89	31.70	421.42	0.55
tiling&extract_bands	2560	388.94	0.49	4	70	3.06	31.16	421.44	0.46
tiling&extract_bands	2560	388.94	0.60	5	70	3.07	31.22	421.82	0.47
VIBSD_inference		6.00				1.58	28.14	52.00	1.31

3.4.2. Simulation Results of an Exemplary Scenario

The objective of the experiment is to evaluate and compare the performance of two scenarios: traditional smoke detection without AI onboard and smoke detection with AI onboard. The input image for the experiment has dimensions of 3072 by 1856 and contains 45 bands. Using the developed emulation system, we measure the runtime, memory footprint, and power consumption associated with each step of the process workflow. By summarizing the performance differences observed between these two scenarios, we gain insights into the potential benefits and efficiencies of integrating AI onboard for smoke detection.

Figure 3-4 presents the simulation results of the onboard process for traditional smoke detection without AI onboard. In this scenario, the captured image is first compressed using LZMA method, resulting in a compressed file size of 370 MB. The compressed image is then split into 778 files, each with a size of 512 KB, ready for downlinking to the ground station. The smoke detection occurs on-ground after the downlinking.

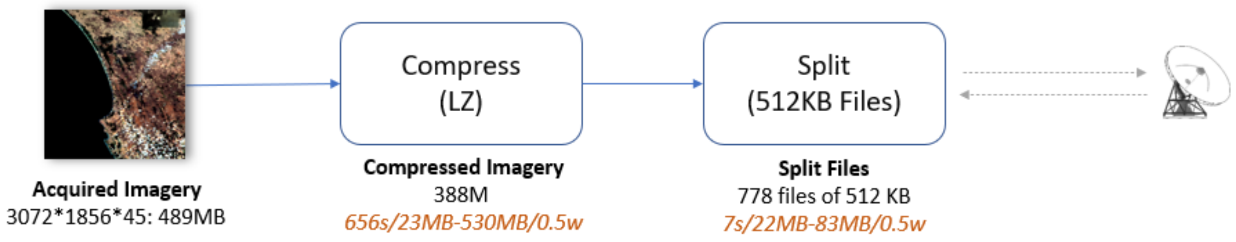


FIGURE 3-4. SIMULATION OF THE ONBOARD PROCESS FOR TRADITIONAL SMOKE DETECTION (WITHOUT AI ONBOARD).

The smoke detection onboard scenario is presented in Figure 3-5. The tiling and band extraction step produces 84 tiles, each with a dimension of 256 by 256 pixels and 3 selected bands, resulting in a tile size of 0.38 MB. After applying the AI model inference, ten tiles containing smoke are detected, which are then merged back into an image with identical dimensions, bands, and georeferencing information as the original imagery. The merged imagery is compressed from 489 MB to 61 MB and split into 122 small files, each with a size of 512 Kilo Byte, for downlinking. Among all steps, compression is the most time and power-consuming operation.

Comparing the two scenarios, the AI onboard scenario consumed 0.142 Watt-hours of power and produced 122 files for downlinking. In contrast, without AI onboard, the power consumption for onboard processing is 0.117 Watt-hours but with 778 files for downlinking. The downlinking data rate of Kanyini is 28.7 Mbps, and assuming the downlink power is 13.7 Watt (derived from the Φ-Sat-1 mission [14]), the power consumption for data downlinking in the AI onboard scenario is 0.065 Watt-hours, while that of the AI on-ground scenario is 0.414 Watt-hours.

In summary, the experiments show that the AI onboard approach for fire smoke detection outperforms the AI on-ground alternative. AI onboard significantly reduces the data downlink volume to just 16% of its original size, resulting in a 69% decrease in energy consumption. In addition, AI onboard detects fire smoke 500 times faster than AI on-ground (see Table 7).
To put this time saving into a real-world fire response scenario, the satellite would need to have a real-time data link to a ground warning station.

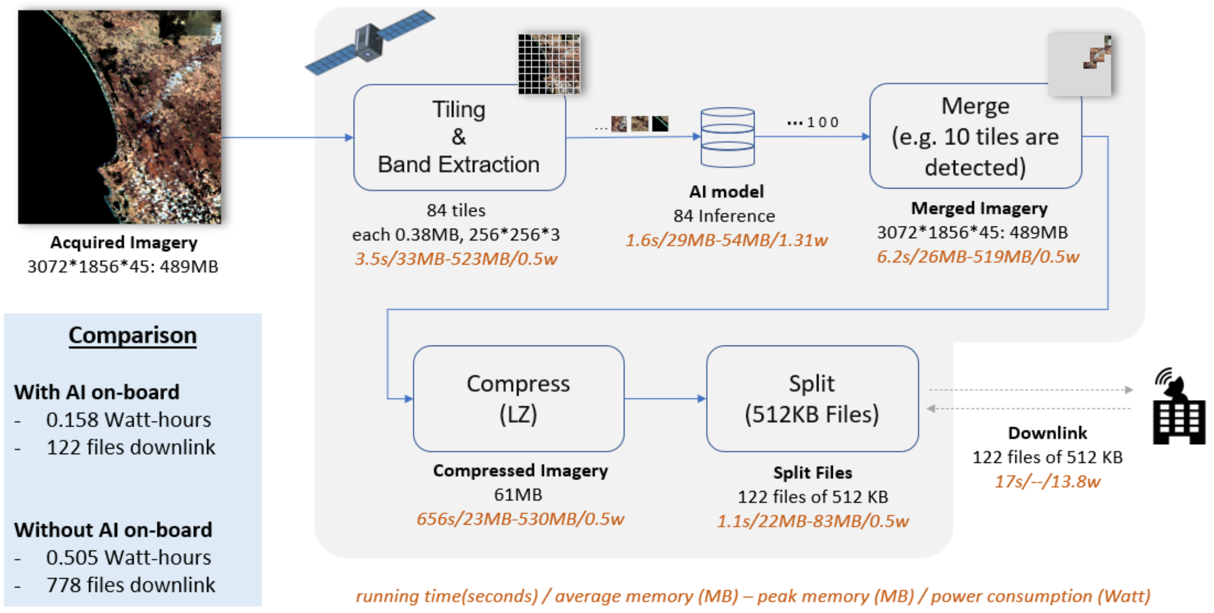


FIGURE 3-5. SIMULATION OF THE ONBOARD PROCESS FOR SMOKE DETECTION ONBOARD.

TABLE 7. AI ON-GROUND VS AI ON-BOARD PERFORMANCE COMPARISON FOR USED TEST SCENARIOS

	AI On-ground	AI On-board	Performance Gain
Detection Time (Seconds)	2571	5	504 times faster
Energy Consumption (Watt-hours)	0.505	0.158	69% less
Data Downlink Requirement (MB)	370	58	84% smaller

3.4.3. All Simulation Results

Simulation experiments were conducted using various specifications, including different numbers of pixels across the track for visual and near-infrared bands (selected from 3072 and 2560) and different numbers of bands (selected from 45 and 48). The results of each step of the onboard processing with these different configurations are presented in the file *emulation_result.csv* in GitHub (*access can be provided on request*).

3.5. Discussions

The accuracy of simulation results is affected by many factors, including differences in hardware, software, and even coding style. For example, the emulation system uses the Myriad X chip from NCS2 for the AI module, while the Kanyini satellite uses the Myriad 2 chip. Therefore, it is recommended to calibrate the emulation system to align with the satellite`s computing environment once access to the satellite or its engineering model becomes available.

3.6. Chapter Summary

The successful development of an emulation system has allowed for the simulation of the Kanyini satellite`s computing environment, facilitating comprehensive performance measurements of various onboard processes. Extensive experimentation has been carried out to evaluate the performance aspects, including running time, memory usage, and power consumption, specifically for fire smoke detection both on-ground and onboard.

The onboard AI approach for fire smoke detection outperforms the AI on-ground alternative. Our experiments show that AI onboard significantly reduces the data downlink volume to just 16% of its original size, resulting in a 69% decrease in energy consumption. In addition, AI onboard detects fire smoke 500 times faster than AI on-ground.

It is worth noting that although the emulation system is configured to match the Kanyini satellite, it is designed to be adaptable and easily applicable to other missions.

3.7. Code

Code and simulation results are available at:

https://github.com/stefanpeters/SmartSat_P2.38/

(*access can be provided on request*)

4. Band Selection

4.1. Motivation and Aims

The HyperScout-2 sensor employed in the Kanyini satellite captures images comprising 45 visible and near-infrared (VNIR) bands and 3 thermal bands. However, when implementing AI onboard for fire smoke detection, the satellite does not have the capability to support a large AI model that utilizes input from all 48 bands. To address this limitation, we conduct band selection on-ground to identify the bands that have the most significant impact on distinguishing between smoke and non-smoke tiles. Only the identified bands are extracted from the tiles during AI inference onboard before being fed into the AI model.

Band selection is performed using three different methods: Pearson correlation, Principal Component Analysis (PCA), and Random Forest (RF). PCA is an unsupervised method that does not require the use of labels. On the other hand, correlation and RF are supervised methods that involve labels for band selection. The experiments are conducted using the simulated hyperspectral imagery described in Section 2.1.

4.2. Data Preparation

The data preparation process consists of three steps: tile cleaning, tile labeling, and tile sampling. The original tiles used are from the simulated tiles generated from VIIRS, as described in Section 2.1. After the tile cleaning step, we obtained a total of 19,955 tiles from various regions, including ACT, Western Australia, and New South Wales. Each tile has a dimension of 256 by 256 pixels and consists of 49 bands, which include 45 VNIR bands, 3 thermal bands, and a mask band. The mask band contains pixel-level labels, assigning each pixel to one of four classes: 0 for clear, 1 for smoke, 2 for mixed, and 3 for cloud.

Figure 4-1 illustrates the distribution of the percentage of pixels in each class across the 19,955 tiles. The X-axis represents the percentage of pixels within a specific class, and the Y-axis represents the number of tiles. Focusing on the distribution of smoke pixels, the subfigure in the upper right corner shows that 2,717 tiles have no smoke pixels and approximately 83% of the tiles (16,567 out of 19,955) contain less than 10% of smoke pixels. Overall, 98% of the tiles have less than 40% smoke pixels.

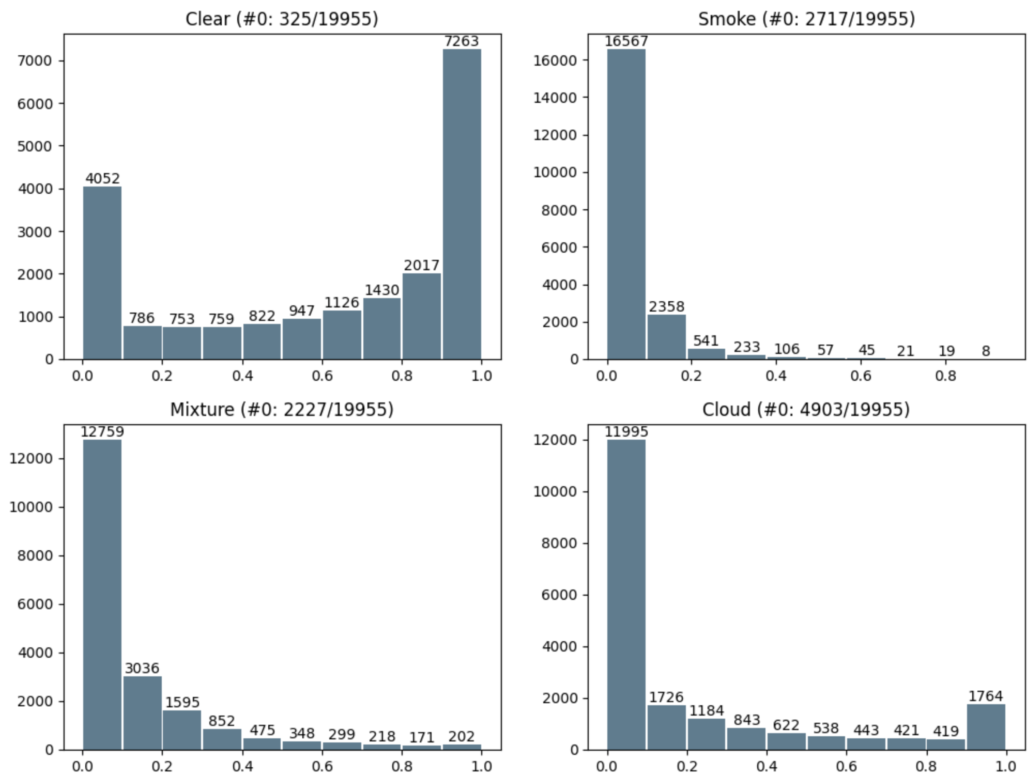


FIGURE 4-1. DISTRIBUTION OF THE PERCENTAGE OF PIXELS IN EACH CLASS OF 19955 TILES

As the AI model, VIB_SD (described in Chapter 5), works to predict whether a tile contains smoke, the pixel-level label needs to be converted into a tile-level label. The tile-level labels consist of two classes: smoke and non-smoke. If a tile contains any smoke pixels, it is categorized as smoke. Otherwise, it is identified as non-smoke. After tile labelling, the smoke and non-smoke tiles distribution are presented in Figure 4-2 and in Figure 4-3.

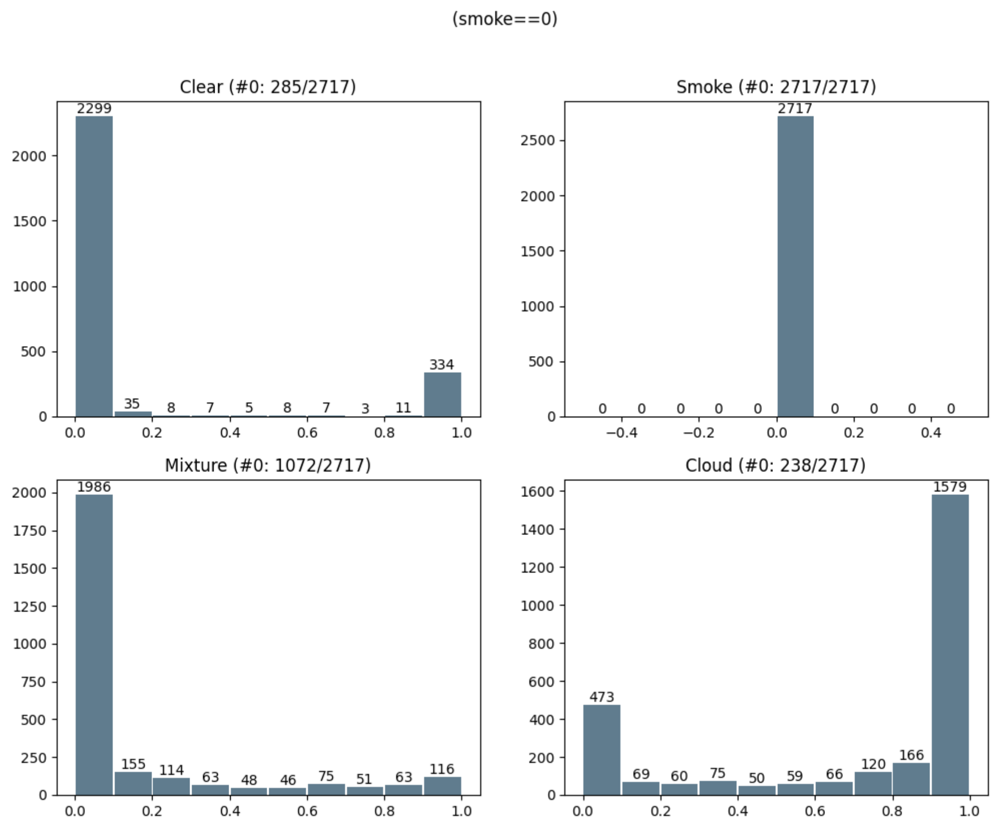


FIGURE 4-2. DISTRIBUTION OF 2,717 NON-SMOKE TILES.

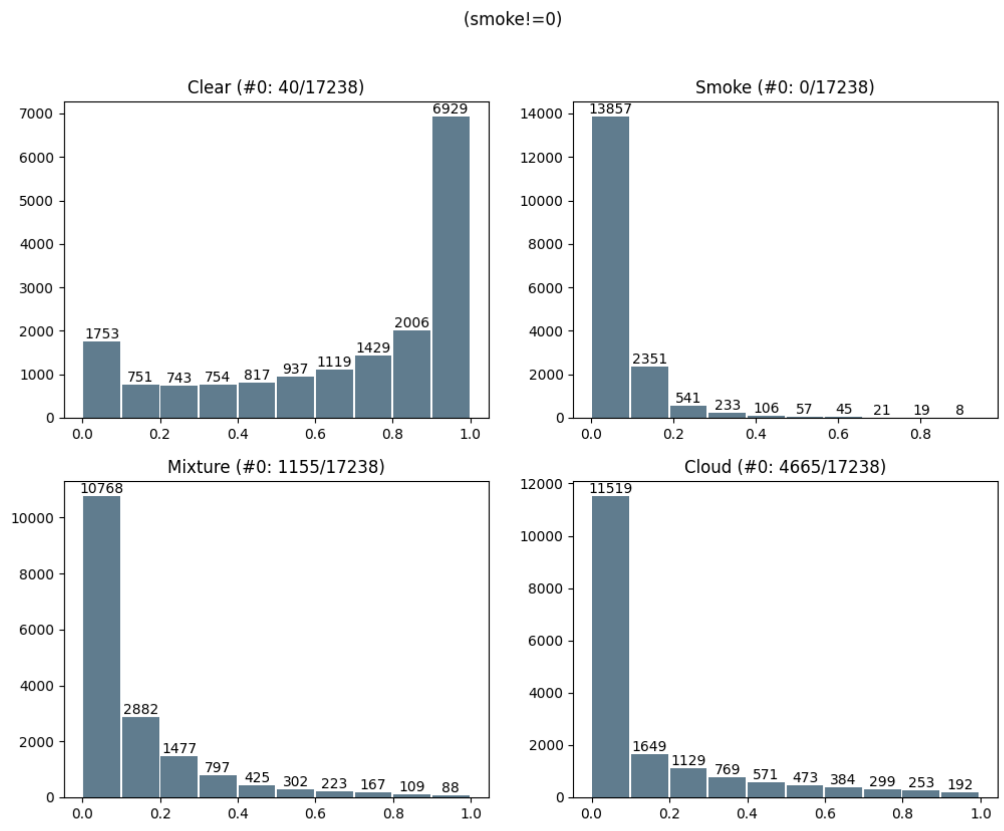


FIGURE 4-3. DISTRIBUTION OF 17,238 SMOKE TILES.

Next, we proceed with sampling 1600 tiles, comprising 800 tiles from each class. The sampling approach aims to create a more balanced distribution of smoke and non-smoke tiles to ensure coverage of different scenarios and reduce potential bias in the data set. The distribution of 800 sampled smoke and non-smoke tiles are presented in Figure 4-4 and in Figure 4-5, respectively. The 800 tiles in the smoke class are used as the dataset for band selection.

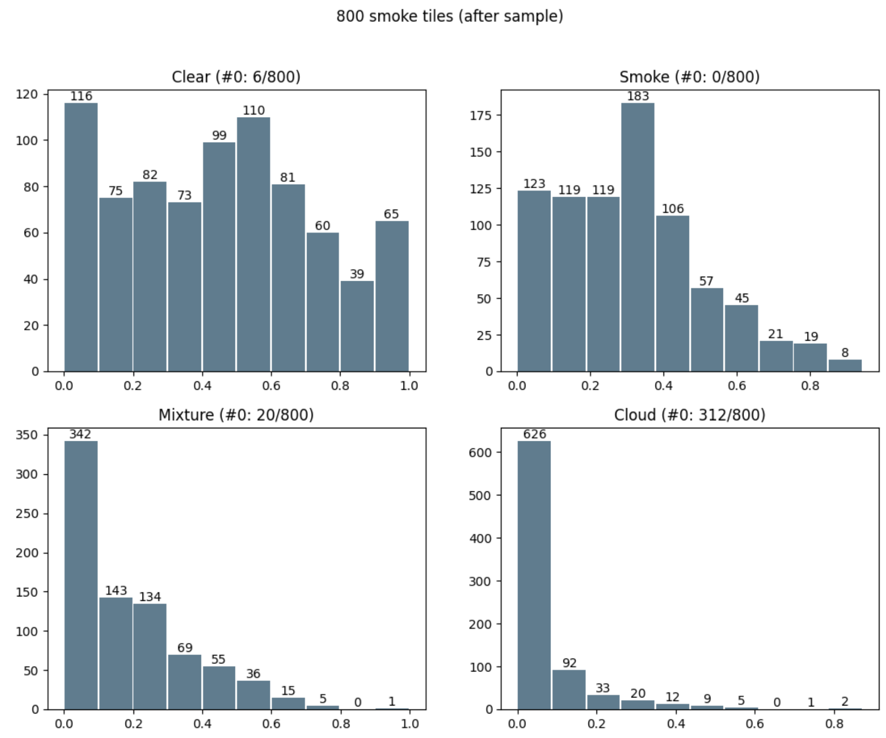


FIGURE 4-4. DISTRIBUTION OF 800 SAMPLED SMOKE TILES

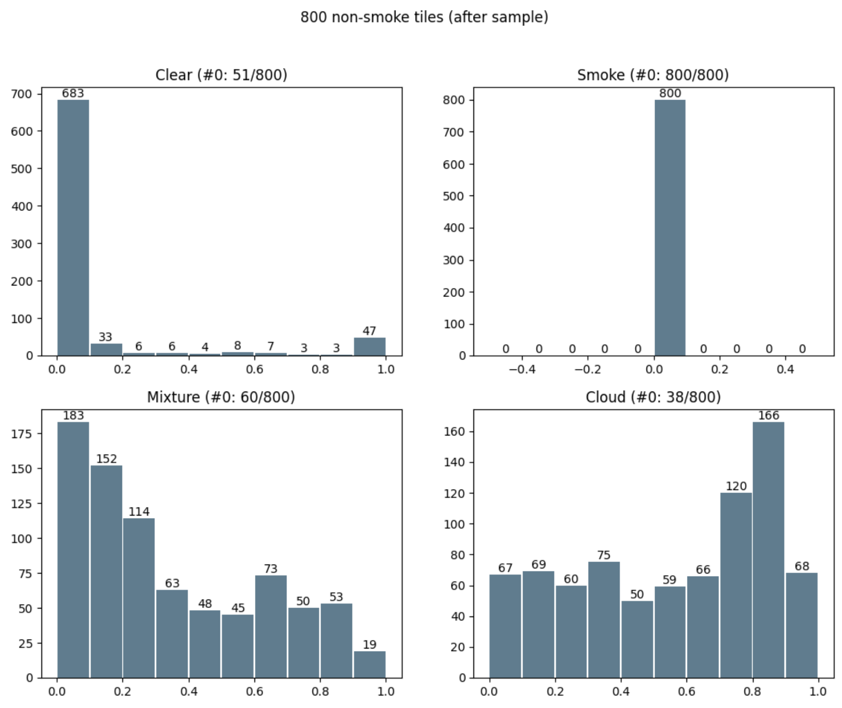


FIGURE 4-5. DISTRIBUTION OF 800 SAMPLED NON-SMOKE TILES

4.3. Band Selection

4.3.1. Band Importance using Pearson Correlation

To compute band importance using Pearson Correlation, each tile with dimensions of 256 by 256 and 49 bands is transformed into a matrix of size 65,536 by 49. Then, the Pearson correlation coefficient is calculated for each tile. The average of the absolute values of the correlation coefficients across the 1600 tiles is computed and presented in Figure 4-6.

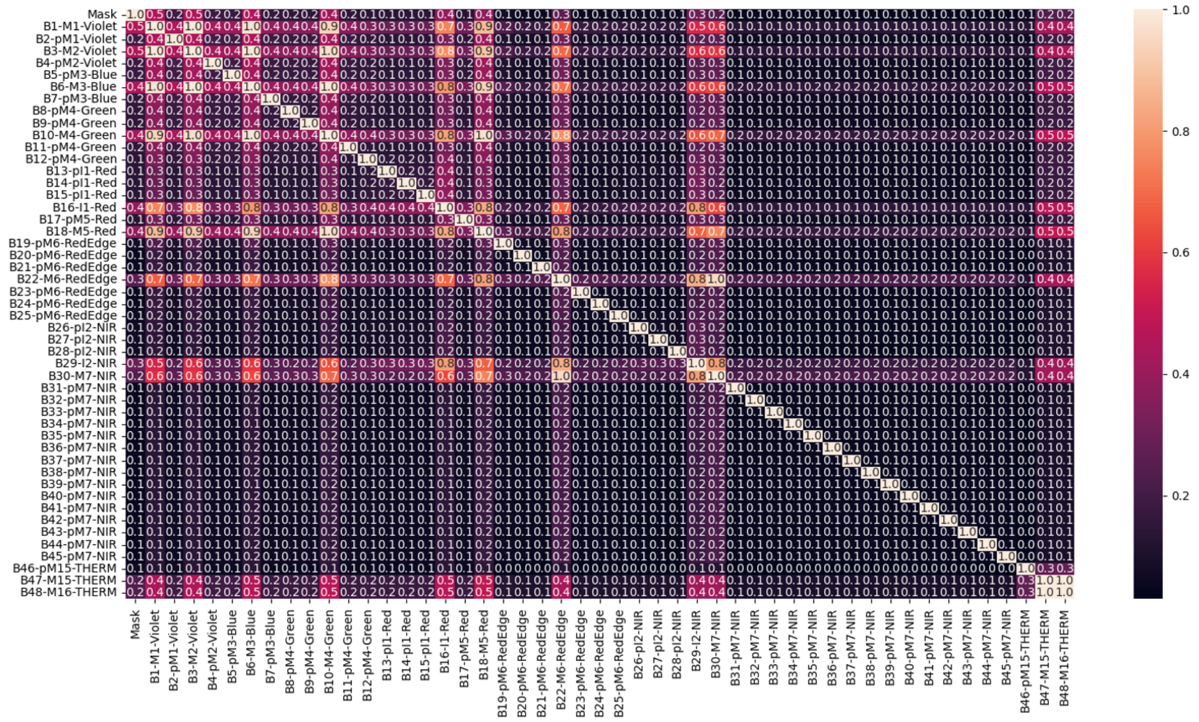


FIGURE 4-6. AVERAGE ABSOLUTE CORRELATION COEFFICIENT OVER THE 1,600 TILES

The important bands are determined based on the highest average correlation with the mask band. According to the correlation method, the top 10 bands are identified as follows: B3-M2-Violet, B1-M1-Violet, B6-M3-Blue, B10-M4-Green, B16-I1-Red, B18-M5-Red, B29-I2-NIR, B22-M6-RedEdge, B47-M15-THERM, and B48-M16-THERM.

4.3.2. Band Importance using Principal Component Analysis

The concept behind Principal Component Analysis (PCA) is to identify the top bands or features that contribute the most to reconstructing the original image. In our experiments, we utilize a variant of PCA known as weighted PCA (wPCA) to calculate the band importance. The steps involved in wPCA are as follows:

- Step 1: For each tile, compute the principal components (PCs).
- Step 2: For each of the top three PCs with the highest percentages to explain the total variance, calculate the absolute value of the weighted coefficients. These coefficients are obtained by multiplying the absolute values of the coefficients with the percentage of explained variance.
- Step 3: Determine the average of the absolute values of the weighted coefficients for the top three PCs, which will serve as the band importance.
- Step 4: Repeat steps 1 to 3 for all tiles to calculate the band importance across the entire dataset.
- Step 5: Use the average value of the band importance from all tiles as the final band importance.
-

The algorithm for obtaining the band importance using wPCA is presented in Appendix B.6 (confidential).

We only consider the top three PCs with the highest explained variance in the computation because they account for the majority of the total variance. As shown in Figure 4-7, the top three PCs explain 23%, 5%, and 3% of the total variance, respectively. In contrast, the remaining PCs exhibit a similar low-variance explanation. Therefore, by considering the top three PCs, we can capture the most significant sources of variation in the dataset.

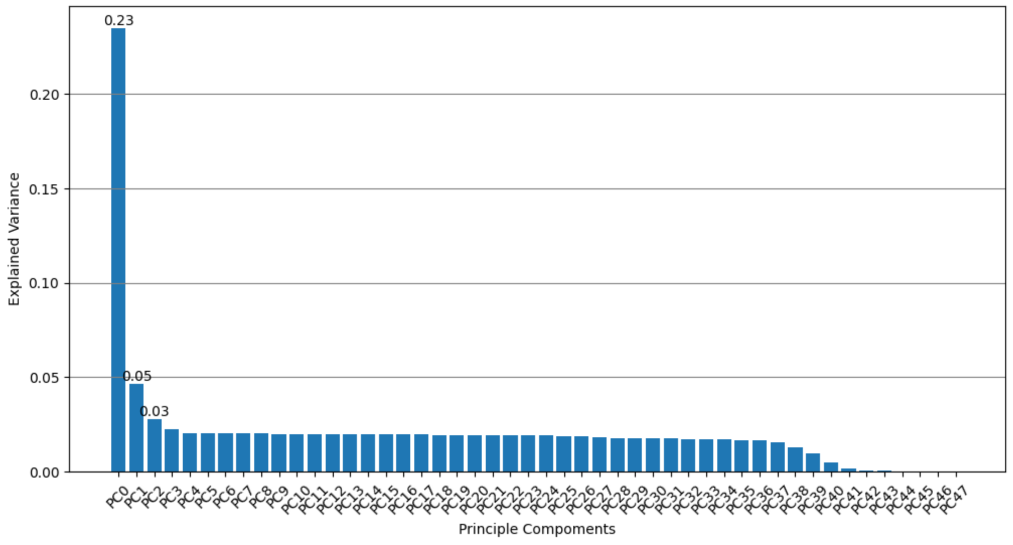


FIGURE 4-7. AVERAGE PERCENTAGES OF VARIANCE EXPLAINED BY 48 PCS.

Figure 4-8 shows the importance of the 48 bands determined by wPCA. Eleven bands exhibit significantly higher importance than the other. These 11 bands are highlighted within the red rectangle in Figure 7-8. It is worth mentioning that the images used in our study are derived from VIIRS imagery, with 11 bands being the original bands and the remaining 37 bands being simulated. It is interesting that wPCA successfully identifies the 11 original bands as more important. This finding implies the effectiveness of the wPCA in discovering the important bands within the dataset.

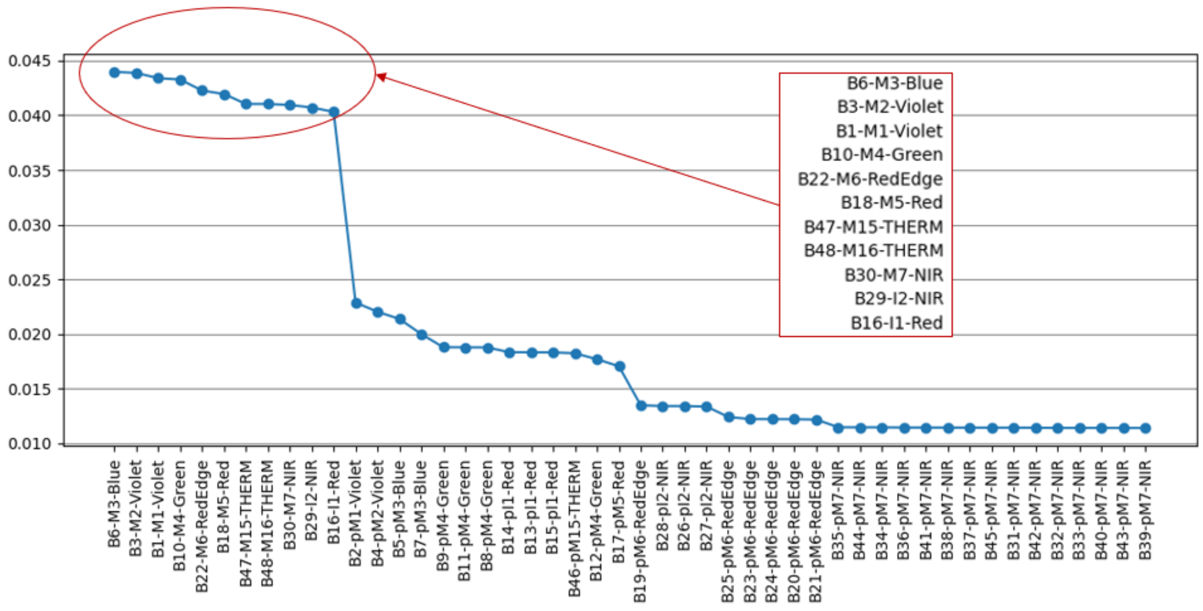


FIGURE 4-8. BAND IMPORTANCE PRODUCED BY WPCA.

4.3.3. Band Importance using Random Forrest

Random forest (RF) is a supervised machine learning algorithm that builds multiple decision trees to make predictions. In the context of band selection, RF is employed to identify the bands that have the most significant impact on predicting the target band, which is the masking band, using information from other bands. In our experiments, we consider three variants of RF: the original RF, RF with permutation (RFP), and RF with Shapley value (RFS).

The three Random Forest (RF) model types have slight differences and unique advantages. The original RF model builds an ensemble of decision trees based on bootstrap aggregating and feature randomization, providing robustness and accurate predictions. RF with permutation evaluates the importance of each feature by permuting its values and measuring the resulting decrease in predictive accuracy, offering a straightforward and reliable assessment. On the other hand, RF with Shapley value calculates feature importance using cooperative game theory principles, capturing the contribution of each feature in combination with others, enabling a more comprehensive understanding of feature interactions.

The band selection process using the three RF methods involves the following steps:

- Step 1: Each tile is transformed into a matrix of dimensions 65,536 by 48, with the mask band separated to serve as the target variable.
- Step 2: The tile is then split into a training set, which comprises 75% of the 65,536 records, and a test set, which accounts for the remaining 25%.
- Step 3: Three RF models are trained using the training set: the original RF model (RF), RF with permutation (RFP), and RF with Shapley value (RFS).
- Step 4: The band importance is determined using the trained RF models.
- Step 5: Steps 1 to 4 are repeated for all tiles in the dataset to calculate the band importance across the dataset.
- Step 6: The final band importance is obtained by taking the average value of the band importances from all tiles.

The band importance results obtained from the three RF models are illustrated in Figure 4-9. All three models successfully identify the 11 original bands as important bands for smoke detection. Importantly, the band importance results generated by the three models exhibit remarkable similarity, indicating consistent and reliable outcomes across the different RF methods.

As RF, RFP and RFS generate similar results, we only use the results of RF in the subsequent analysis and experiments.

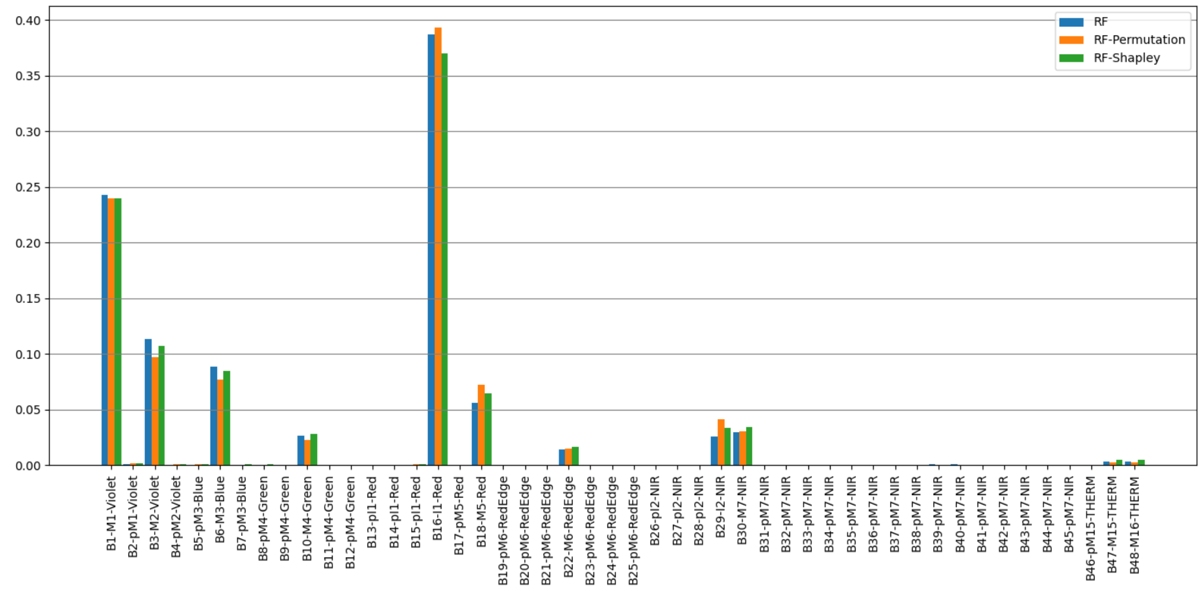


FIGURE 4-9. BAND IMPORTANCE OBTAINED BY RF, RFP AND RFS.

4.3.4. Summary of Band Selection Results

Figure 4-10 summarizes the band importance results obtained from the correlation, PCA, and RF methods. All three methods demonstrate consistent findings by identifying the 11 original bands as important bands. Notably, the RF method assigns higher importance to B16-I1-Red than the other two methods while marking the two thermal bands, B47 and B48, with relatively lower importance.

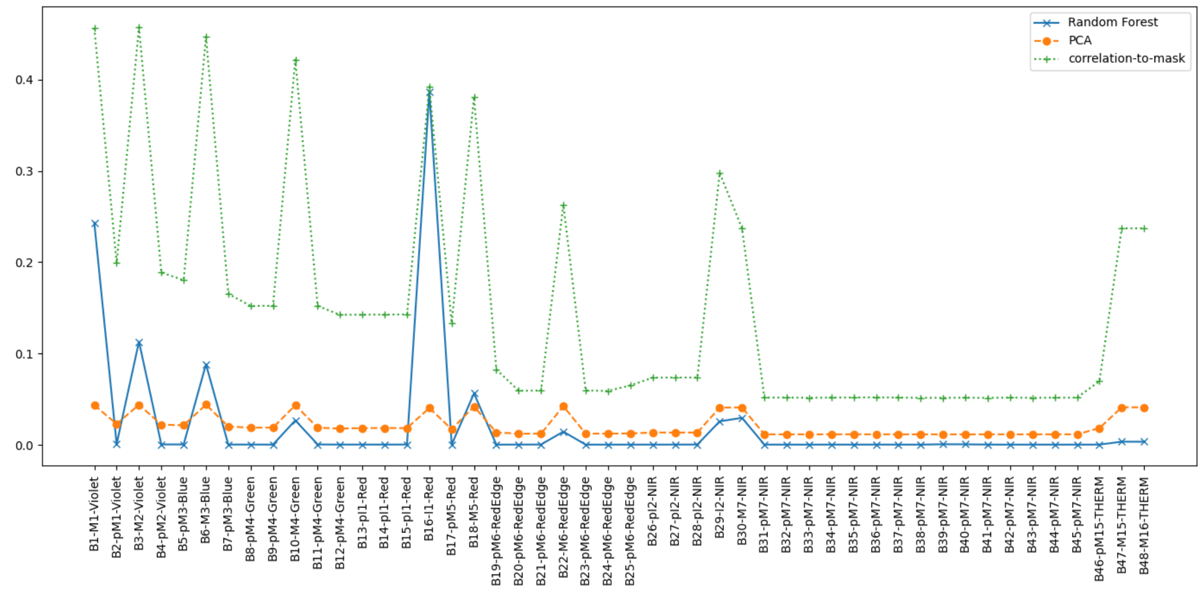


FIGURE 4-10. BAND IMPORTANCE PRODUCED BY CORRELATION, PCA AND RF.

As highly correlated bands are redundant in a prediction model, we filter out the bands with an absolute value of the Pearson correlation coefficient larger than 0.95 among the top 11 important bands identified by the three methods. The resulting top important bands for each method are as follows:

- RF: B16, B1, B18, B30, B29
- PCA: B6, B22, B47, B29, B16
- Correlation: B3, B16, B29, B22

4.4. Smoke Detection with Selected Bands

To evaluate the smoke detection performance using the selected bands, we conducted experiments on datasets comprising different sets of bands. In addition to the three band selection methods discussed earlier (RF, PCA, and Correlation), we included two additional methods: Ground Truth (GT) bands and bands with low importance (LBI).

In the GT method, the selected bands were chosen based on the ground truth labels. As described in Chapter 6, bands B16, B22, B29, B1, B6, and B30, are used to label the mask band, which are selected for GT. For the LBI method, we selected bands consistently identified as low importance by all three band selection methods. The selected bands for LBI include B31, B32, B33, B34, B35, and B36. A summary of the selected bands of the five methods is presented in Table 8.

TABLE 8. SELECTED BANDS BY FIVE DIFFERENT METHODS.

	RF	PCA	COR	GT	LBI
Top 1	B16-I1-Red	B6-M3-Blue	B3-M2-Violet	B16-I1-Red	B31-pM7-NIR
Top 2	B1-M1-Violet	B22-M6-RedEdge	B16-I1-Red	B22-M6-RedEdge	B32-pM7-NIR
Top 3	B18-M5-Red	B47-M15-THERM	B29-I2-NIR	B29-I2-NIR	B33-pM7-NIR
Top 4	B30-M7-NIR	B29-I2-NIR	B22-M6-RedEdge	B1-M1-Violet	B34-pM7-NIR
Top 5	B29-I2-NIR	B16-I1-Red		B6-M3-Blue	B35-pM7-NIR
Top 6				B30-M7-NIR	B36-pM7-NIR

From the five band selection methods, we generated 21 sets of bands by selecting a range of 2 to 6 bands from each of the methods (if available). Using these band sets, we created corresponding datasets by selecting the specified bands from the sampled 1600 tiles. The dataset is named the band selection method, followed by the number of bands selected. For example, COR4 refers to the dataset generated by selecting the top 4 bands of the correlation method. Then, each dataset was randomly divided into a training set (60% of the tiles), a validation set (20% of the tiles), and a test set (20% of the tiles). We trained the VIB_SD model (described in Chapter 5) using the training and validation sets and used the trained model to predict the test set. This process was repeated 10 times for each dataset, resulting in 210 trained models and 210 prediction results. From the 10 prediction results obtained for each dataset, we report the average, minimum, maximum, and standard deviation.

To evaluate the prediction performance, we measured the accuracy, true positive rate (TPR), false positive rate (FPR), true negative rate (TNR), and false negative rate (FNR) for each prediction. Among these measurements, we particularly focus on accuracy and

FNR, representing the overall prediction performance and the rate of missed detections, respectively.

The experimental results are presented in Table 9, in which the best results are in bold fonts. Among the two comparison methods, LBI exhibits the lowest prediction accuracy, highlighting the importance of selecting more important bands. On the other hand, the three band selection methods (RF, PCA, and correlation) yield comparable accuracy to the ground truth (GT). Among the three band selection methods, COR3, COR4 and RF4 achieve the highest accuracy of 95.7% with an FNR of 2.8%, 2.3% and 2.1%, respectively. Overall, selecting 4 top bands yields good performance across all three methods, while COR3 performs similarly but produces smaller models. Increasing the number of selected bands did not necessarily improve accuracy.

TABLE 9. SMOKE DETECTION ACCURACIES RESULTS USING VIB_SD WITH DIFFERENT SELECTED BANDS OBTAINED BY RF, PCA, AND CORRELATION.

Band Selection Method	#Bands	Accuracy	False Negative Rate
COR	2	0.897 ± 0.013	0.068 ± 0.019
GT	2	0.912 ± 0.016	0.038 ± 0.020
LBI	2	0.550 ± 0.020	0.245 ± 0.091
PCA	2	0.912 ± 0.016	0.035 ± 0.016
RF	2	0.890 ± 0.020	0.071± 0.019
COR	3	**0.957** ± 0.010	0.028 ± 0.014
GT	3	0.925 ± 0.016	0.031± 0.013
LBI	3	0.555 ± 0.032	0.277 ± 0.085
PCA	3	0.939 ± 0.012	0.030 ± 0.013
RF	3	0.919 ± 0.024	0.042 ± 0.014
COR	4	**0.957** ± 0.008	0.023 ± 0.011
GT	4	**0.957** ± 0.009	**0.021** ± 0.011
LBI	4	0.582 ± 0.025	0.248 ± 0.077
PCA	4	0.943 ± 0.009	0.027 ± 0.006
RF	4	**0.957** ± 0.010	**0.021** ± 0.010
LBI	5	0.547 ± 0.023	0.218 ± 0.087
PCA	5	0.955 ± 0.011	0.023 ± 0.014
RF	5	0.953 ± 0.019	0.032 ± 0.015
GT	6	0.950 ± 0.012	0.023 ± 0.009
LBI	6	0.560 ± 0.049	0.183 ± 0.115

The prediction accuracy for the different numbers of selected bands using the five methods is illustrated in Figure 4-11. Among the two comparison methods, LBI exhibits the lowest prediction accuracy, highlighting the importance of selecting more important bands. On the other hand, the three band selection methods (RF, PCA, and correlation) yield comparable accuracy to the GT method, suggesting their effectiveness in capturing relevant information for accurate predictions.

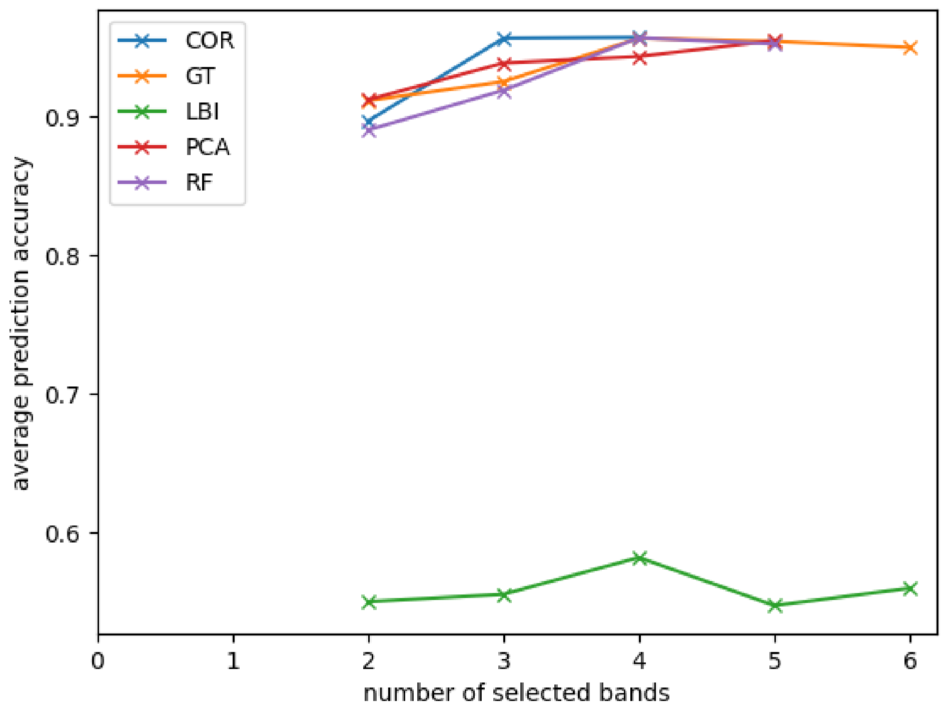
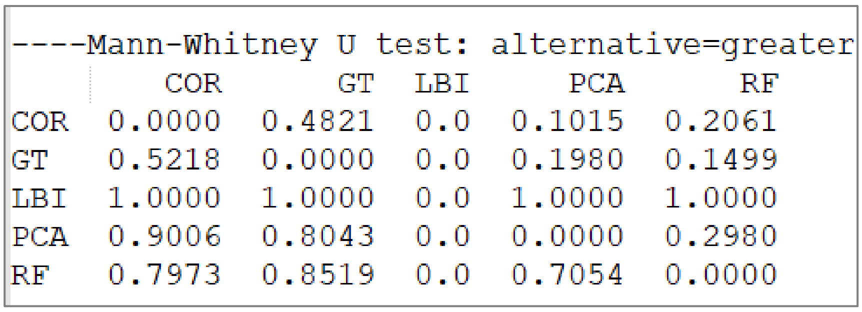


FIGURE 4-11. PREDICTION ACCURACY WITH DIFFERENT NUMBERS OF SELECTED BANDS OF THE FIVE METHODS.

We conduct Mann-Whitney U tests pairwise of the three methods to determine if one method is statistically better than another method. The null hypothesis assumes that the accuracy distribution of one method is stochastically greater than the distribution of another method. The p-values of the Mann-Whitney U tests are presented in Table 10. Although COR shows slightly better accuracy than PCA and RF, the differences are not statistically significant.

TABLE 10. THE P VALUES OF THE MANN-WHITNEY U TEST.



```
----Mann-Whitney U test: alternative=greater
         COR      GT    LBI     PCA      RF
COR  0.0000  0.4821  0.0  0.1015  0.2061
GT   0.5218  0.0000  0.0  0.1980  0.1499
LBI  1.0000  1.0000  0.0  1.0000  1.0000
PCA  0.9006  0.8043  0.0  0.0000  0.2980
RF   0.7973  0.8519  0.0  0.7054  0.0000
```

Considering the prediction performance and model size, the best band set is from COR3, corresponding to B3, B16, and B29. COR4 and RF4 demonstrate similar prediction performance.

4.5. Discussions

Although the primary goal of band selection is to identify bands that have a significant impact on distinguishing smoke from non-smoke image tiles, there are minor differences in the specific goals of the three methods. Correlation aims to identify bands that show a high linear correlation with smoke/non-smoke labels. PCA aims to identify the bands that contribute most to the reconstruction of the original image. On the other hand, RF aims to determine which bands contribute most to classifying a pixel as smoke or non-smoke. Despite these differences in goals, the bands identified as highly important by the three methods tend to exhibit a considerable degree of similarity.

4.6. Chapter Summary

In this chapter, we explore band selection using three different methods: correlation, PCA, and RF, on simulated HyperScout-2 images sourced from the VIIRS dataset. All three methods effectively identify the 11 original bands of the VIIRS imagery, highlighting their effectiveness. We conduct extensive experiments to evaluate smoke prediction accuracy using the VIB_SD model on datasets created with the selected bands. Our results show that by selecting three specific bands (B3, B16, and B29), we achieve the highest prediction accuracy of 95.7% with a false negative rate (FNR) of 2.3%.

It is worth noting that in this phase, our focus was not on the prediction accuracy of AI models. Further efforts can be undertaken to validate the prediction performance on more complex datasets, investigate different AI models, and fine-tune the model to enhance its effectiveness and efficiency for onboard inference.

4.7. Code

Code and simulation results are available at:

https://github.com/ShaLu-ML/SS-P2.38

(*access can be provided on request*)

5. Uplink and Downlink Data Transfer

5.1. Motivation and Aims

With Kanyini capturing hyperspectral imagery and processing these onboard using AI, compressed raw imagery or/and onboard imagery processing results will be downlinked to a ground station. There are currently two ground stations that will be used, one located at the North Pole (Svalbard) and one at the South Pole (Troll). Conversely, the AI model files can be uplinked to the satellite for an updated version of the onboard AI smoke detection algorithm.

This section aims to estimate the time required in various settings for data transfer. Since the satellite is not yet launched and the engineering model is not accessible at the stage either, data transfer is simulated using a software package developed in-house.

The specific goals for this chapter include:

- Develop a software framework for simulating the data transfer process.
- Run the simulation to estimate the transfer time required for various scenarios.

5.2. Methods and implementation

First, the design of the simulation framework is presented. Key parameters can be found at the "Constants" section and implementation details will be shown with code examples. After the explanation of the design, the specific setups will be presented to simulate various scenarios. These are referred to as experiments. Each experiment aims to answer one or two very specific questions, such as how the transfer speed influences the transfer time.

5.3. Simulation framework

The framework borrows the idea of molecular dynamics in computational chemistry, where the state of the system is calculated (or initiated at the beginning) at a given time. The state in this case comprises a group of descriptors for the satellite, including the orbital elements, its current location, heath status of the hardware, etc. The following text will use "State" (uppercase S) when referring to this concept.

The idea using the State is to perform data transfer in a stepwise approach. At the given State, if the satellite is within the communication range with the ground station, a packet of data is considered sent. The size of this packet is determined by the transfer speed and the time interval, which are pre-defined for each simulation. The State is then updated to the next time frame and this process continues until the file has been completely sent. The entire process is similar to trapezoidal rule in calculus to approximate an area using the sum of multiple trapezoids.

The following section introduces the State implementation in detail.

5.4. State implementation

5.4.1. Essential orbital elements

First and foremost, the essential orbital elements of Kanyini are encoded in the “State”. These are a subset of the generic orbital elements (Figure 5-1). Default values are selected for certain parameters, which reduces the number of independent variables.

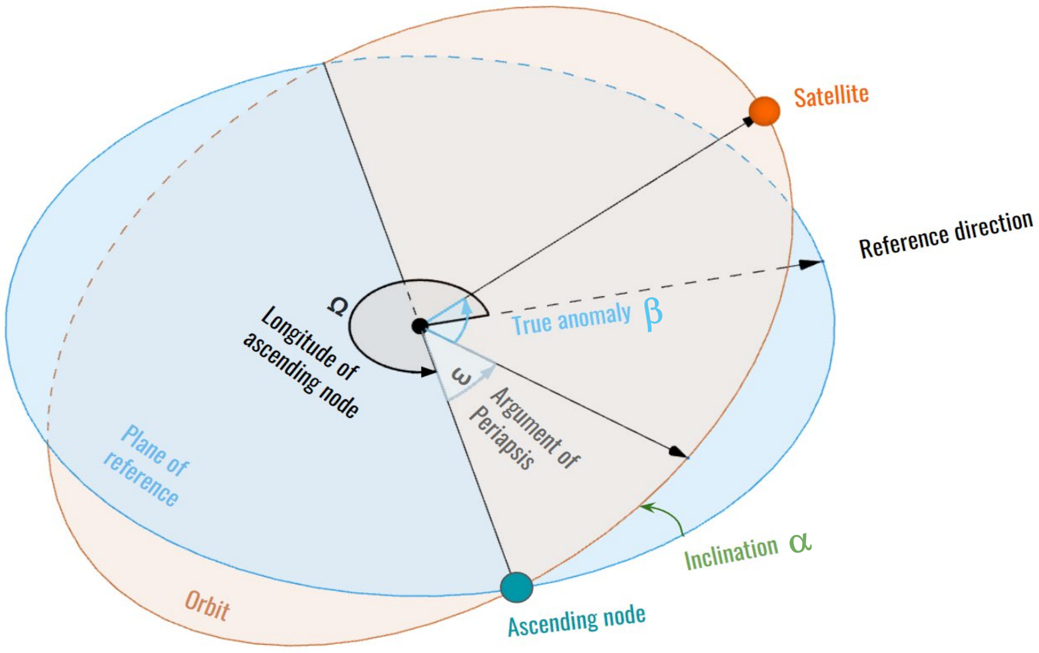


FIGURE 5-1: GENERIC ORBITAL ELEMENTS

Default values are chosen for:

- Reference direction: centre to 0°E
- Longitude of ascending node: 0
- Plane of reference: 0°N (the equator)

This leads to the Ascending node to be (0°N, 0°E). Now, we set the Argument of periapsis to 0, the two independent variables left are the Inclination and the True anomaly. To determine the exact position, altitude is also needed. Hence in the simulation, the position of the satellite is deterministically specified using the three parameters:

- alpha: value for the Inclination, 0 - 1/2 π.
- beta: value for the True anomaly, 0 - 2 π
- altitude: value for altitude, 500 - 550[12] (km)

[12] Technical Kanyini/HyperScout-2 details provided by Nick Manser, Satellite Systems Engineer, SmartSat-CRC

During a simulation, the alpha and altitude are mostly fixed, beta is the key parameter to determine the satellite location. It's the main parameter to update between frames.

5.4.2. Other parameters

To enable the simulation, the following parameters are included in a State.

- period: period of Kanyini. This determines how much distance the satellite will move in a given time;
- com_state: communication state. This determines whether the satellite is within range of at least one ground station.
- dev_state: device state. This determines whether the hardware for data transfer is functional. It's used to simulate the stochastic interruptions (e.g., by space radiation) on the hardware. This report assumes the satellite is healthy throughout the transfer period.
- phi_offset: the difference between the reference direction and 0°E. This is to include the relative movement between of the Earth and the reference direction. As the Earth rotates, this value decreases.
- beta_inc: whether beta increments or decrements in the next frame. This is to control the direction of the satellite movement. For alpha not in the range of [0, ½π], changing this parameter can achieve the equivalent orbit.

5.4.3. Updating the State

The main component to update in a State is beta, which influences the satellite location. This is done by specifying a time interval (delta_t) for the satellite to travel and calculating the angular difference of beta.

In the implementation, the State has a function move() to update the satellite to the next position by specifying delta_t. The function will check the direction of the satellite, specified by beta_inc, modify the beta and finally include the effect of the Earth rotation, as is shown in Appendix B.7 (confidential).

The altitude of Kanyini is provided as a range (500 – 550 km). To simulate the change in the altitude, the framework includes a mechanism to enable the satellite float between the specified altitude range.

5.4.4. Running the simulation

The design of running a simulation borrows the idea from PyTorch, which combines the ease of working with a data model (in this case, the State) and customisation (the user specifies each step).

The steps of performing a simulation include:

- Define the configurations such as the file size, the transfer speed, the communication range, and ground stations to communicate with.
- Initialise the State with a location. This can be a random or specifically picked location.
- Enter a loop to transfer and update the satellite position, until the data packet sent exceeds the file size.
- Check whether the satellite is in range.
- If it's in range, send a packet of data.
- Update the satellite position.

As an option, the user can specify the health of the device and update the altitude in the data transfer loop. Appendix B.8 (confidential) provides an excerpt from the code to demonstrate the process in which the total time is calculated at the end.

5.4.5. Logs

For each simulation, the information at every time frame can be saved, known as a State record.

5.5. Experiments

Various scenarios are considered to evaluate factors that influence the transfer time.

5.5.1. Experiment 1 - duration in range with the ground stations

This is to get the distribution of duration when Kanyini is in the communication range with the ground stations. Two grounds are assessed:

- Svalbard for the North Pole
- Troll for the South Pole

This also functions as a quality assurance of the framework as the Kanyini engineering model team has provided an independent source of results to compare against.

5.5.2. Experiment 2.1 - downlink time vs file size

This is to evaluate the effect of file size on the final time required. The file size ranges from 0 to the standard size of 500 Mb using the speed DownLink1 28.7 Mb/s. This is to simulate only a portion of the image being sent back, as AI compression reduces the file size. Three configurations are used for the ground stations:

- Svalbard only
- Troll only
- Svalbard and Troll

5.5.3. Experiment 2.2 - downlink time vs transfer speed

This is to evaluate the effect of transfer speed on the downlink time. The simulation will transfer 500 Mb using different transfer speeds. The ground station setting is the same as Experiment 2.1.

5.5.4. Experiment 2.3 - transfer within Australia

Previous simulations assume the satellite can be at any position. It’s of more interest to estimate the time when the data transfer initiates inside Australia. Firstly, a grid of points is sampled in the Australia continent. Then each point is used as the initial position to start the simulation. The border information is obtained from the World Administrative Boundaries[13].

5.5.5. Experiment 2.4 - time to uplink model files

This is to estimate the time to send files from the ground station to the satellite for updating the AI model. The setup is similar to Experiment 2.1, except the file size (model size: 8 Mb) and transfer speed (56 kb/s) is different.

[13] https://public.opendatasoft.com/explore/dataset/world-administrative-boundaries/export/

5.6. Constants

Constants used in the simulations are shown in Table 11.

TABLE 11: CONSTANTS USED FOR THE SIMULATION.

Category	Item	Value	Unit
Ground stations	Svalbard (North Pole)		
	Latitude	78.23	(km)
	Longitude	15.41	
	Altitude	0.45	
	Troll (South Pole)		
	Latitude	-72.02	
	Longitude	2.53	
	Altitude	0	
Kanyini	Inclination	97.95	(degree)
	Altitude	525	(km)
	Period	96 * 60	(s)
Earth	Radius	6378.1	(km)
	Angular velocity	2 π / 3600 / 24	(rad / s)
File transfer	Communication range	1800	(km)
	DownLink1	28.7 * 2**10	(kb / s)
	DownLink2	12.5 * 2**10	(kb / s)
	UpLink1	56	(kb / s)
File size	Standard downlink file size	500 * 2**10	(kb)
	Model uplink size	8 * 2**10	(kb)

` Note: x**y means x to the power of y.

5.7. Results

5.7.1. Experiment 1 - duration in range with the ground stations

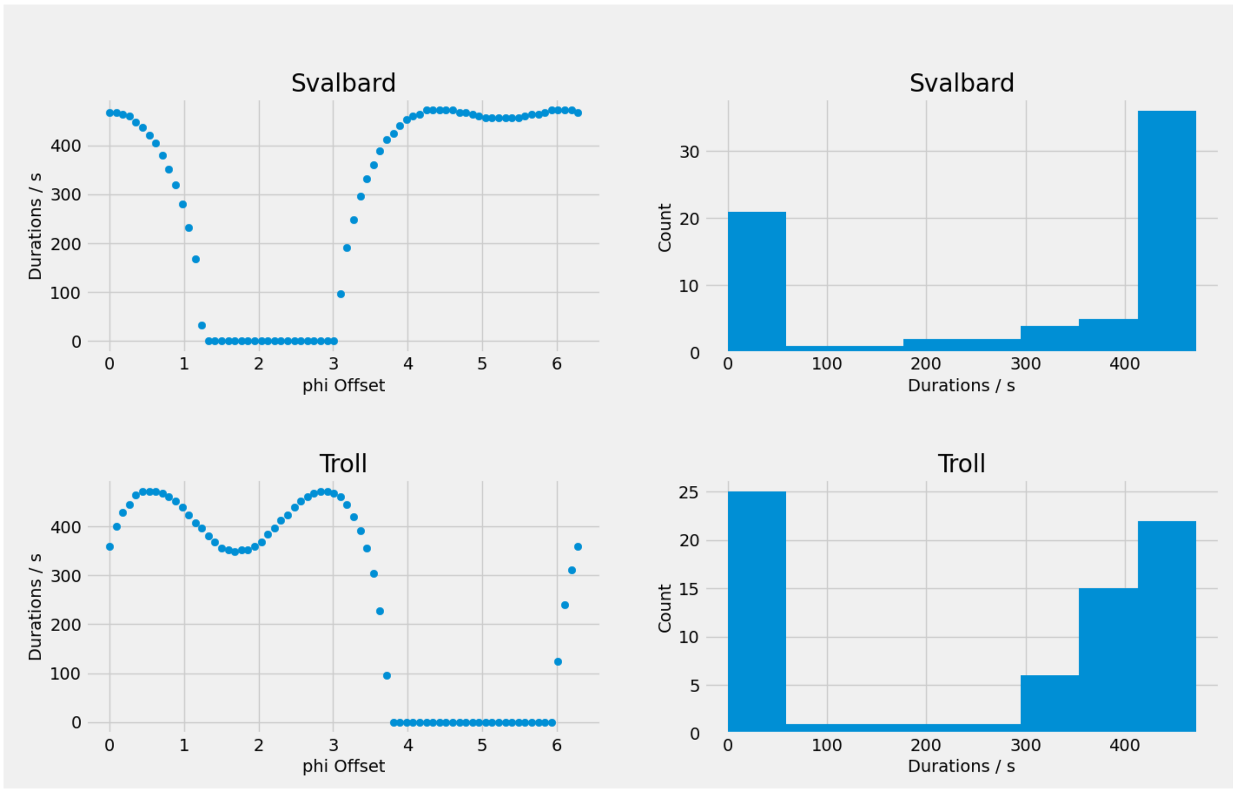


FIGURE 5-2: DURATION WHEN THE SATELLITE IS IN RANGE WITH THE GROUND STATION. (PHI OFFSET: THE ANGLE DIFFERENCE BETWEEN THE REFERENCE DIRECTION AND 0°E)

Results of the durations for either ground stations are shown in Figure 5-2. In the left-hand scatter plots, each point corresponds to one cycle of the Kanyini orbit. The phi offset (0 - 2 π) indicates the initial position of the simulation, i.e., the longitude of the starting position. The duration in the communication period is dependent on the starting longitude. Noticeably, some positions end up with no communication with the ground. The right-hand side histograms are on the same data. If the zeros are excluded, the simulation aligns well with the technical details provided by the Kanyini team (Table 12).

TABLE 12: TOTAL COMMUNICATION TIME FROM THE SIMULATION AND THE TEAM

Item	Simulation (min)	Simulation excluding zeros (min)	From engineering model team (min)

Troll	4.3 ± 3.3	6.5 ± 1.4	6.5
Svalbard	4.9 ± 3.4	6.75 ± 1.7	6.8

5.7.2. Experiment 2.1 - downlink time vs file size

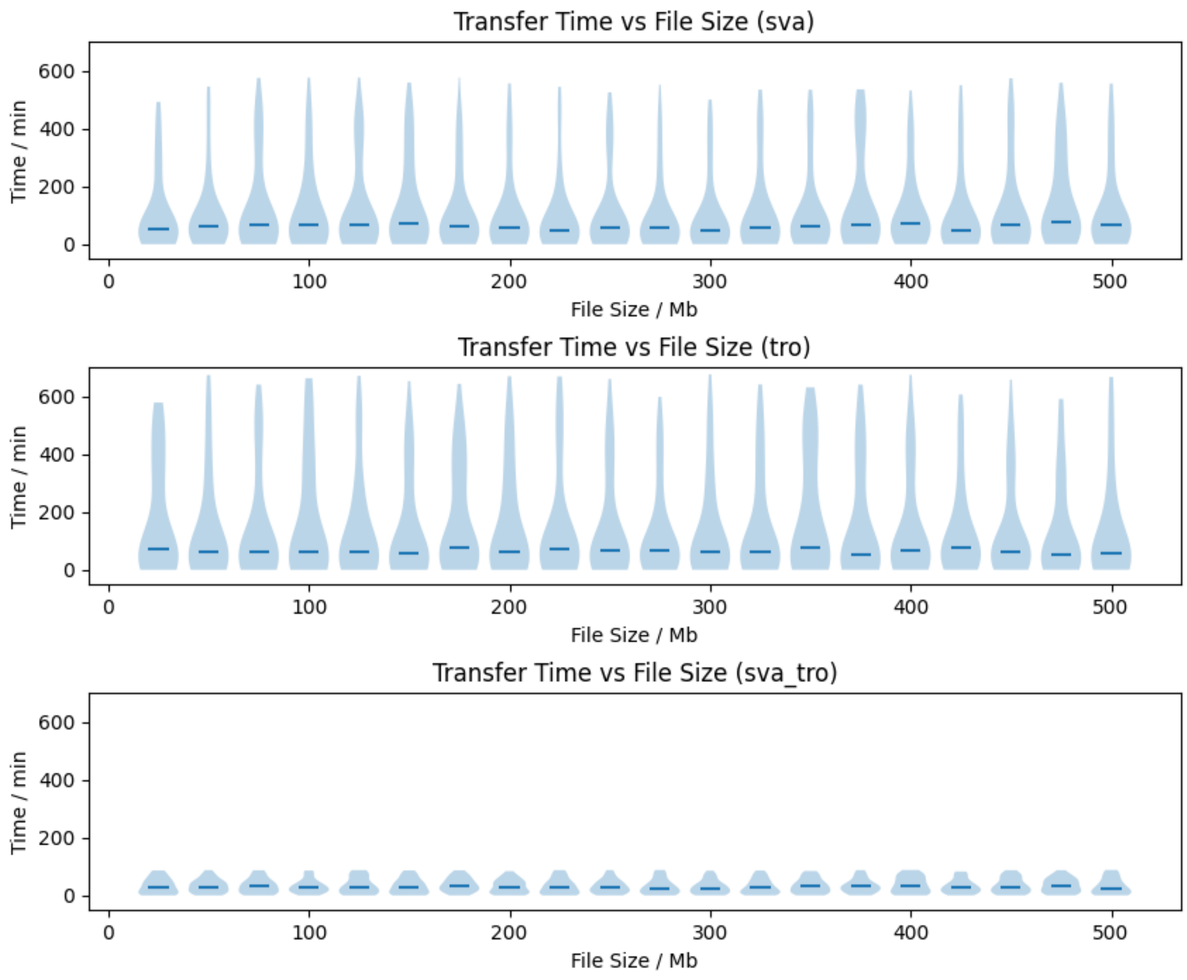


FIGURE 5-3: TRANSFER TIME FOR VARIOUS FILE SIZES. THE SIMULATION IS TO DOWNLINK VARIOUS PORTIONS OF A 500 MB FILE USING A TRANSFER SPEED OF 28.7 MB/S. THREE GROUND SETUPS ARE USED INCLUDING SVALBARD ONLY (SVA), TROLL ONLY (TRO) AND BOTH (SVA_TRO). THE SUBPLOTS SHARE THE SAME Y AXES.

The results presented in Figure 5-3 show that using one ground station (either Svalbard or Troll), a long tail is presented for the transfer time. Using Svalbard alone, the median transfer time is 3875 seconds for 500 Mb which is not much larger than for 2.5 Mb (3200 seconds). In comparison, using both stations together significantly speed up the transfer, with only 1510 seconds for 500 Mb. The long tail is also absent when both grounds are in use, indicating the transfer time is much more stable and less variant to the initial position.

5.7.3. Experiment 2.2 - downlink time vs transfer speed

Figure 5-4 and Table 13 show the results for downlink time using various transfer speeds. The time required to downlink is relatively stable for a speed between 4 to 28.7 Mb/s However, the time increases markedly when the speed is below 4 Mb /s, especially below 1 Mb/s. This pattern is found in all ground station settings. However, using both ground stations will also dramatically lower the amount of time spent for all speeds.

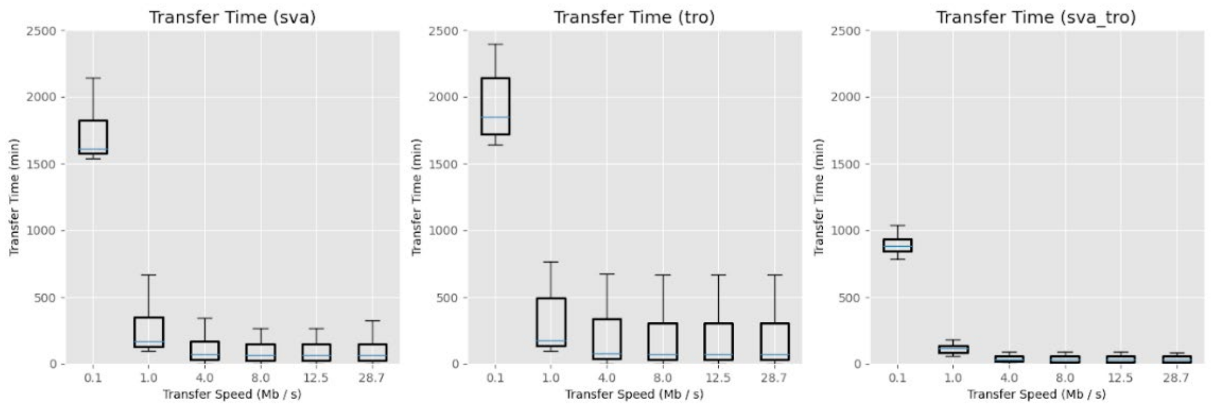


FIGURE 5-4: DOWNLINK TIME FOR 500 MB USING DIFFERENT TRANSFER SPEEDS. THREE GROUND SETUPS ARE USED INCLUDING SVALBARD ONLY (SVA), TROLL ONLY (TRO) AND BOTH (SVA_TRO). THE SUBPLOTS SHARE THE SAME X AND Y AXES.

TABLE 13: TIME TO TRANSFER 500 MB AT DIFFERENT SPEEDS

Ground Station	**Time / minutes (mean ± standard deviation)**				
	0.1 Mb/s	1.0 Mb/s	4.0 Mb/s	12.5 Mb/s	28.7 Mb/s
Svalbard	1717.8 ± 189.8	256.7 ± 175.0	130.6 ± 148.8	121.7 ± 138.7	120.2 ± 137.0
Troll	1929.7 ± 232.9	299.1 ± 211.6	180.4 ± 193.6	167.7 ± 184.9	165.4 ± 183.3
Both	894.2 ± 64.7	114.4 ± 35.1	37.8 ± 25.4	36.0 ± 25.6	34.6 ± 25.3

5.7.4. Experiment 2.3 - transfer within Australia

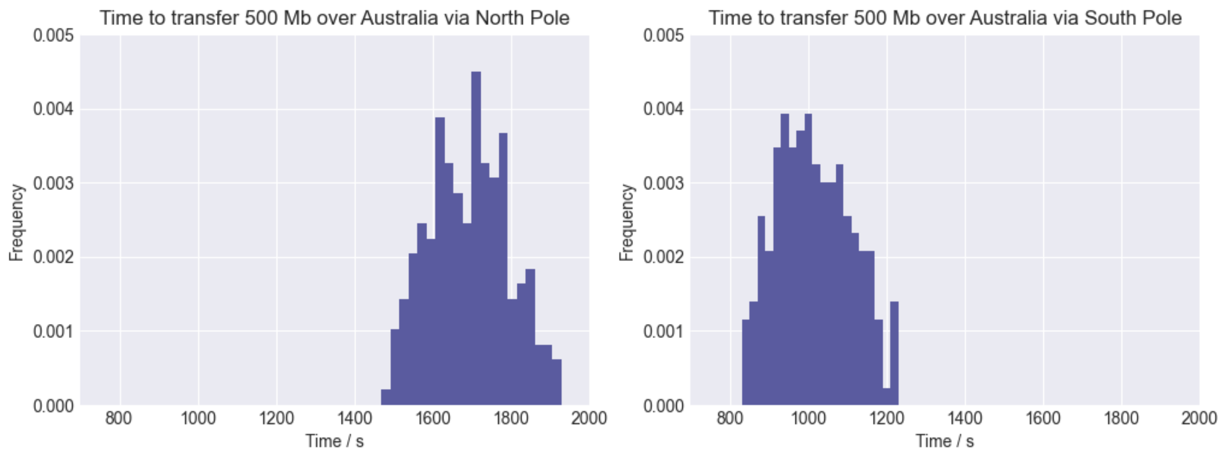


FIGURE 5-5: TIME REQUIRED TO TRANSFER 500 MB STARTING IN AUSTRALIA. TRANSFER CAN BE VIA THE GROUND IN THE NORTH POLE (LEFT) OR THE SOUTH POLE (RIGHT). THE SUBPLOTS SHARE THE X AND Y AXES.

Two scenarios are provided to simulate the satellite passing Australia. The first scenario is that the satellite will pass Australia from south to north and reach the North Pole ground station (Figure 5-5, left). The other scenario is the opposite, moving from south to north and transferring data with the South Pole station (Figure 5-5, right).

Based on the information provided[14], using the South Pole is the actual setup for Kanyini. In that case, since the South Pole is close to Australia, the average time to finish data transfer (1009.4 ± 95.2 s), which is approximately 16.7 minutes. This is shorter than passing the North Pole (1693.6 ± 101.2 s). Figure 5-6 shows the scatter plot of the transfer time in both scenarios. Since points are sampled in Australia, the points resemble the Australia continent shape.

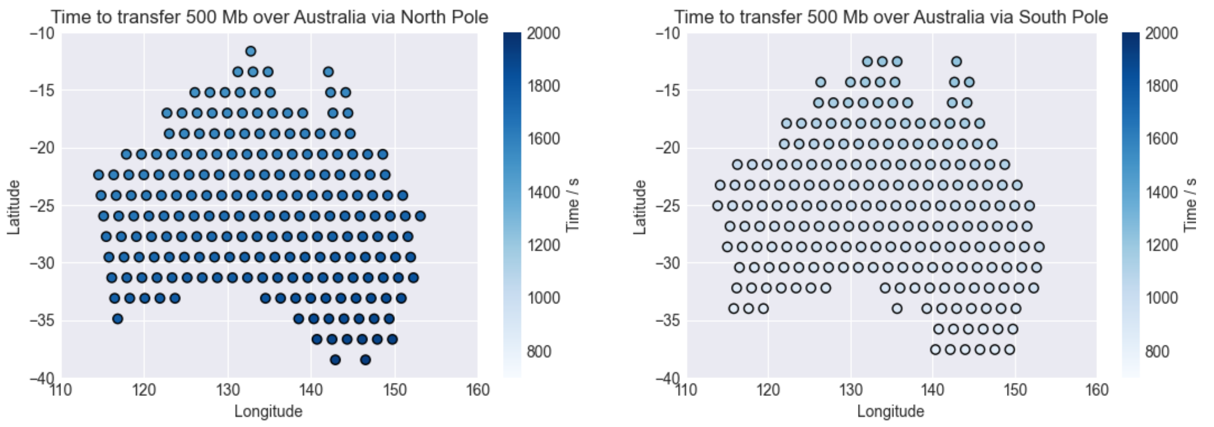

Time to transfer 500 Mb over Australia via North Pole
Time to transfer 500 Mb over Australia via South Pole
Latitude
Longitude
Time / s
−10
−15
−20
−25
−30
−35
−40
110
120
130
140
150
160
2000
1800
1600
1400
1200
1000
800

FIGURE 5-6: GRID POINTS SHOWING THE TIME REQUIRED TO TRANSFER 500 MB STARTING IN AUSTRALIA. TRANSFER CAN BE VIA THE GROUND IN THE NORTH POLE (LEFT) OR THE SOUTH POLE (RIGHT). THE SUBPLOTS SHARE THE X AND Y AXES.

[14] Technical Kanyini/HyperScout-2 details provided by Nick Manser, Satellite Systems Engineer, SmartSat-CRC

Applying our simulated data transfer model (experiment 2.3) to our demonstration example – the fire event in the Coorong, SA (-36.653361, 140.187622), the estimated mean data transfer time is 855 seconds with a standard deviation of 17.32 seconds. In other words, it takes 14 minutes and 15 seconds from the moment Kanyini-HS2 has captured imagery over the fire event until Kanyini has reached (moving southwards) the communication zone over the South Pole and the onboard processed data have been completed to be downlinked to the ground station.

If Kanyini was moving north, estimated mean and standard deviation data transfer time would be 1870.0 and 18.26 seconds respectively.

5.7.5. Experiment 2.4 - time to uplink model files

The results for uplink (Figure 5-7) are quite similar to the downlink counterpart. Using both ground stations greatly speed up the transfer. For the maximum case (8 Mb), it takes 7419.1± 8414.0 s for using Svalbard alone and 9223.0 ± 10811.9 s for Troll alone. This number is 2210.9 ± 1670.1 s to uplink using both stations, which is about 37 minutes on average.

5.8. Discussions

First, it’s important to notice that Kanyini can stay out of the communication zone with the ground station when passing over it for certain longitudes. In Figure 5-2, there are zones of approximately two radians for either Svalbard or Troll ground station. Fortunately, the two zones at the poles do not overlap. This means if both ground stations are used, the satellite will at least communicate with one station, which has already considered the Earth’s rotation during the satellite travel time from one pole to another.

When using only one ground station, there is a long tail in the transfer time for both downlink (Figure 5-3) and uplink (Figure 5-7). This is because the satellite may need to travel for certain hours before reaching the communication range of a ground station. For downlink, since the transfer speed is relatively fast, most transfers can be completed within one pass near the station. Hence the time to reach the ground becomes the main component for the total transfer time. Therefore, the time is relatively invariant to the file size (Figure 5-3), causing no significant difference to downlink 100 Mb or 500 Mb. However, when the transfer speed is very low, it requires more than one pass to finish the transfer, causing the waiting time to accumulate (Figure 5-4, at 0.1 Mb/s). Overall, using both ground stations is the key to lowering the transfer time, even at 50% of the standard transfer speed.

Simulating the cases when Kanyini is passing Australia is more related to the final usage, since the algorithm is to detect smoke within Australia. In Figure 5-5, it shows when passing Australia, the satellite will reach either ground stations at one attempt based on the time. This excludes the unfortunate scenario that the satellite stays in the non-communication zone for a long time before the transfer starts. It was indicated that the direction of Kanyini is from north to south over Australia[15]. This makes the data transfer even faster since the South Pole is much closer from Australia than the North Pole. Hence using standard downlink setup (28.7 Mb/s for

[15] Technical Kanyini/HyperScout-2 details provided by Nick Manser, Satellite Systems Engineer, SmartSat-CRC

500 Mb), the transfer time is around 1009 seconds or 6 - 7 minutes when passing Australia – crucial for providing timely information to early warning systems.

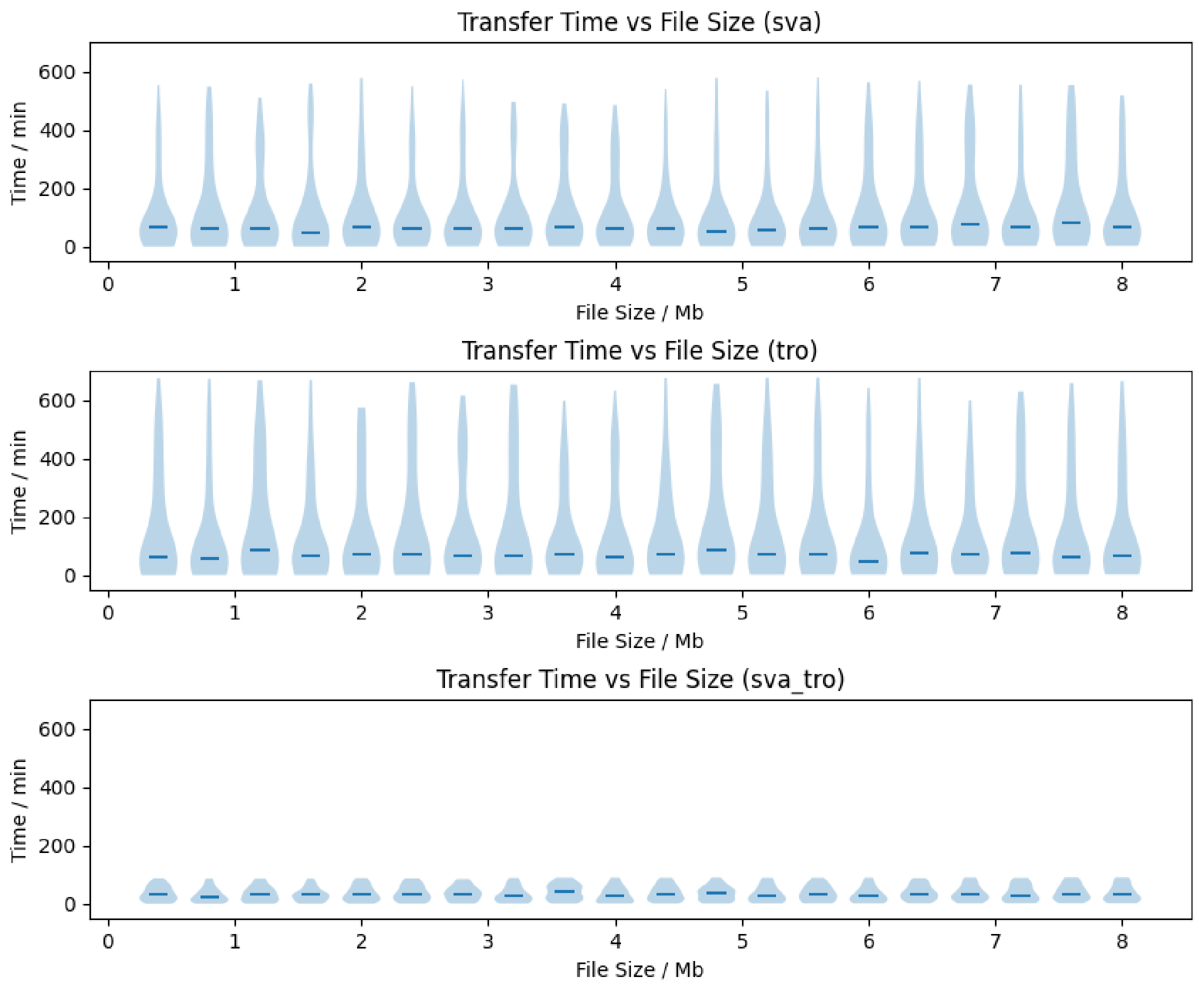


FIGURE 5-7: TRANSFER TIME FOR UPLINKING MODEL FILES. THE SIMULATION IS TO DOWNLINK A MAXIMUM OF 8 MB USING A TRANSFER SPEED OF 56 KB/S. THREE GROUND SETUPS ARE USED INCLUDING SVALBARD ONLY (SVA), TROLL ONLY (TRO) AND BOTH (SVA_TRO). THE SUBPLOTS SHARE THE SAME Y AXES.

Due to the time and scope of this project, limitations do exist, and some solutions are included in the framework. First, the altitude is mostly fixed in the simulation, while in practice it might vary during the flight. The framework includes a mechanism to set the altitude range and update the altitude at each time frame. Second, the communication range is set to 1800 km. In reality, data might still be able to transfer but with a reduced speed. Hence instead of using a hard cut-off, a soft continuous function could be used to handle the range. Third, the simulations in this report assume the satellite hardware is functional throughout the course. However, disruptions from space radiation or low temperature can cause the hardware to malfunction. The framework has a device state built in for this purpose, which can be used to model random disruptions. Lastly, Kanyini has certain power constraints which are not taken into account for the up/downlink simulation. With the constraint details available, the power or energy should be also tracked alongside with the file sent, which adds one more term to the communication criteria.

5.9. Chapter Summary

A framework is successfully developed to estimate data up/downlink transfer time for Kanyini. Quality assurance is performed by checking the communication durations at both ground stations.

For data transfer, using both ground stations is the key factor to reduce the transfer time. Using both ground stations, it takes on average 35 minutes to downlink 500 Mb and 37 minutes to uplink 8Mb using standard speeds (28.7 Mb/s for downlink and 56 k/s for uplink) at arbitrary locations. When the area is limited to Australia, Kanyini takes, on average 1009 seconds (16.8 minutes).

Future work can consider the power and energy constraints to refine the results further.

5.10. Code

Code is available at:

https://github.com/sunyu0410/Kanyini-Data-Transfer

(*access can be provided on request*)

6. Project conclusion and outlook

In conclusion, this research project has provided a promising solution for energy-efficient AI-based onboard processing of hyperspectral imagery for early fire smoke detection, especially for the pre-launch stage. The deployment of the VIB_SD model, operating within the constraints of the HyperScout-2 sensor on the Kanyini satellite, has demonstrated promising results in terms of high prediction accuracy and low false negative rate. The simulation of a comprehensive training dataset and the implementation of an emulation system have further confirmed the feasibility and benefits of onboard processing. The significant reduction in data downlink volume and energy consumption, coupled with the increased speed of fire smoke detection achieved through AI onboard, highlight the practical advantages of onboard AI-based fire smoke detection.

The VIB_SD model adjusted for simulated Kanyini/HyperScout-2 imagery demonstrated high fire smoke prediction accuracy and a low false negative rate while consuming 69% less energy in an onboard scenario compared to a ground-based one. At the same time our AI onboard approach significantly reduced the data downlink volume to just 16% of its original size while running 500 times faster compared to the AI on-ground scenario. The findings of this research not only contribute to advancing satellite-based fire smoke detection capabilities but also hold promise for enhancing wildfire monitoring and response efforts.

Nevertheless, it is crucial to validate its performance using actual Kanyini imagery data once available. Real-world conditions, such as variations in atmospheric conditions, cloud cover, and sensor noise, may affect the model's performance and should be considered for future evaluations. Furthermore, it is also important to note that the accuracy of emulation results is affected by many factors, including differences in hardware, software, and even coding style. For example, the emulation system uses the Myriad X chip from NCS2 for the AI module, while the Kanyini satellite uses the Myriad 2 chip. Therefore, it is recommended to calibrate the emulation system to align with the satellite's computing environment once access to the satellite or its engineering model becomes available.

Future work in the post-launch phase involves calibrating the emulation system to match the actual environment of Kanyini. Additionally, there is a need for model implementation and stepwise model updates using real Kanyini imagery training data after the launch. Exploring alternative AI models, such as segmentation models, can also be considered. Lastly, conducting a comparison between AI-based and deterministic (spectral index-based smoke detection) onboard methods would provide valuable insights. It also needs to be noted that the report did not include assessing the model's performance using manually labelled simulated imagery data. This will be however more important during post-launch phase using manually labelled training data derived from actual Kanyini/HS2 imagery instead of simulated ones.

Another important issue to be discussed is onboard georegistration: The VIIRS imagery products utilised in this project for simulating HyperScout-2 imagery and used as training data for our AI smoke detection algorithm were VIIRS/NPP Surface Reflectance data (VNP09GA and VNP09CMG) – equivalent to L2A. Our focus was primarily on onboard performance testing using simulated Satellite imagery. Therefore, we based our decision for using VIIRS Surface Reflectance products mainly on data availability and ease of image access. The design of our AI onboard algorithms and the conducted performance testing is independent from the choice of imagery input level (L1/L2). However, our current approach depends on the existence of an onboard data handling system (ODHS) {Esposito, 2018

#123} which is capable to “process in real time the acquired L0 data performing geometric corrections and orthorectification”. As of October 2023, we did not yet receive confirmation whether such ODHS will be available on Kanyini, and if so, how it will influence the performance of our methods. Although onboard georegistration and various kinds of onboard corrections could be done using some “default” solutions (“mathematical models”), to maintain the performance we’ve demonstrated, these solutions might need to be “customised” or even “renovated” – likely a larger amount of work ideally conducted before satellite launch with testings on the satellite’s engineering model.

Lastly, looking ahead (remaining pre-launch and post-launch phase), we consider the following research activities:

- Algorithm calibration (assuming given access to the Kanyini engineering model)
- Algorithm advancement (from tile-based solution to segmentation-based pixel-level detection)
- Model updating: Our pre-trained model using simulated data should allow immediate detection of fire smoke assuming the above mentioned ODHS being in place, but the model should be step-wise re-trained/updated using real/new HS2 imagery captured over active fire events to (likely significantly) improve detection accuracy. Once done, further research can be conducted to optimize power consumption and detection accuracy. Retraining might also require re-designing models for different input data, including data collection.
- Exploration of transfer learning strategies using simulated training data and (initially) smaller amounts of newly created Kanyini-HS2-based training data.
- Investigating detection accuracy vs power consumption: To achieve the best possible fire smoke detection accuracy might require more onboard energy. We plan to investigate how detection accuracy is impacted in different available energy scenarios and how onboard atmospheric correction and the use of L1B product will impact energy consumption and detection accuracy compared to using L1C TOA imagery data. This assumes that the Kanyini mission includes an integrate onboard Atmospheric Correction and potentially also onboard Topographic Correction.
- Study the balance and strategies for onboard processing and in-ground processing, i.e. which decisions are made onboard and which decisions are made in-ground and how to achieve them?
- Study the optimal utilisation of onboard storage, i.e. determine what data is transferred to the ground, what data is discarded and how long to keep data in a CubeSat.
- Studying the trade-off between model size (how big is model is), time efficiency, and energy efficiency for onboard processing.
- Investigate alternative deterministic smoke detection methods implemented onboard as an alternative approach to our AI model.
- Investigating further data uplink and downlink solutions targeting performance increase (e.g. data compression; simulating CubeSat constellation scenario).
- Expanding the AI onboard fire smoke detection model towards fire detection and potentially to other applications.

7. References

1. Tedim, F., et al., *Defining extreme wildfire events: Difficulties, challenges, and impacts.* Fire, 2018. **1**(1): p. 9.
2. Borchers Arriagada, N., et al., *Unprecedented smoke‐related health burden associated with the 2019–20 bushfires in eastern Australia.* Medical Journal of Australia, 2020. **213**(6): p. 282-283.
3. Chung, Y.-S. and H. Le, *Detection of forest-fire smoke plumes by satellite imagery.* Atmospheric Environment (1967), 1984. **18**(10): p. 2143-2151.
4. NAGATANI, I. and J.-i. KUDOH, *A False-color Composite Method for Wildfire Smoke Plume Identification Using MODIS Data.* Journal of the remote sensing society of Japan, 2013. **33**(1): p. 38-47.
5. Chen, S., et al., *Global2Salient: Self-adaptive feature aggregation for remote sensing smoke detection.* Neurocomputing, 2021. **466**: p. 202-220.
6. Ba, R., et al., *SmokeNet: Satellite smoke scene detection using convolutional neural network with spatial and channel-wise attention.* Remote Sensing, 2019. **11**(14): p. 1702.
7. Shi, W., et al., *Edge computing: Vision and challenges.* IEEE internet of things journal, 2016. **3**(5): p. 637-646.
8. Zhang, B., et al., *Progress and challenges in intelligent remote sensing satellite systems.* IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2022. **15**: p. 1814-1822.
9. Giuffrida, G., et al., *CloudScout: A deep neural network for on-board cloud detection on hyperspectral images.* Remote Sensing, 2020. **12**(14): p. 2205.
10. Mateo-Garcia, G., et al., *Towards global flood mapping onboard low cost satellites with machine learning.* Scientific reports, 2021. **11**(1): p. 7249.
11. Centre, S.A.S.I. *SA Space Services Mission: SASAT1*. 2023 01.10.2023]; Available from: https://sasic.sa.gov.au/precinctsprojects/sasat1-space-services-mission/.
12. Zhao, L., et al., *Investigating the Impact of Using IR Bands on Early Fire Smoke Detection from Landsat Imagery with a Lightweight CNN Model.* Remote Sensing, 2022. **14**(13): p. 3047.
13. Esposito, M. and A.Z. Marchi. *In-orbit demonstration of the first hyperspectral imager for nanosatellites*. in *International Conference on Space Optics—ICSO 2018*. 2019. SPIE.
14. Giuffrida, G., et al., *The Φ-Sat-1 mission: The first on-board deep neural network demonstrator for satellite earth observation.* IEEE Transactions on Geoscience and Remote Sensing, 2021. **60**: p. 1-14.
15. Gómez, C., J.C. White, and M.A. Wulder, *Optical remotely sensed time series data for land cover classification: A review.* ISPRS Journal of photogrammetry and Remote Sensing, 2016. **116**: p. 55-72.
16. Cai, Y., et al., *A high-performance and in-season classification system of field-level crop types using time-series Landsat data and a machine learning approach.* Remote sensing of environment, 2018. **210**: p. 35-47.
17. Gazzea, M., et al., *Automated power lines vegetation monitoring using high-resolution satellite imagery.* IEEE Transactions on Power Delivery, 2021. **37**(1): p. 308-316.
18. Mutanga, O., T. Dube, and F. Ahmed, *Progress in remote sensing: vegetation monitoring in South Africa.* South African Geographical Journal, 2016. **98**(3): p. 461-471.
19. Olmanson, L.G., P.L. Brezonik, and M.E. Bauer, *Remote sensing for regional lake water quality assessment: capabilities and limitations of current and upcoming satellite systems.* Advances in watershed science and assessment, 2015: p. 111-140.
20. Schumann, G.J., et al., *Assisting flood disaster response with earth observation data and products: A critical assessment.* Remote Sensing, 2018. **10**(8): p. 1230.
21. Boccardo, P. and F. Giulio Tonolo. *Remote sensing role in emergency mapping for disaster response*. in *Engineering Geology for Society and Territory-Volume 5: Urban Geology, Sustainable Planning and Landscape Exploitation*. 2015. Springer.
22. Zhao, S., et al., *The role of satellite remote sensing in mitigating and adapting to global climate change.* Science of The Total Environment, 2023: p. 166820.
23. Yang, J., et al., *The role of satellite remote sensing in climate change studies.* Nature climate change, 2013. **3**(10): p. 875-883.
24. Munoz-Martin, J.F., et al., *In-orbit validation of the FMPL-2 instrument—The GNSS-R and L-band microwave radiometer payload of the FSSCat mission.* Remote Sensing, 2020. **13**(1): p. 121.

25. Camps, A., et al. *FSSCat mission description and first scientific results of the FMPL-2 onboard 3CAT-5/A*. in *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*. 2021. IEEE.
26. Lavender, S., *Technical Note on Quality Assessment for FSSCat HyperScout-2.* Earthnet Data Assessment Pilot, 2022. **EDAP.REP.044**: p. 1-25.
27. Syed, M.H., et al., *Addressing image and Poisson noise deconvolution problem using deep learning approaches.* Computational Intelligence, 2022.
28. Zhu, Z. and C.E. Woodcock, *Automated cloud, cloud shadow, and snow detection in multitemporal Landsat data: An algorithm designed specifically for monitoring land cover change.* Remote Sensing of Environment, 2014. **152**: p. 217-234.
29. Frantz, D., et al., *Improvement of the Fmask algorithm for Sentinel-2 images: Separating clouds from bright surfaces based on parallax effects.* Remote Sensing of Environment, 2018. **215**: p. 471-481.
30. Buchard, V., et al., *Using the OMI aerosol index and absorption aerosol optical depth to evaluate the NASA MERRA Aerosol Reanalysis.* Atmospheric Chemistry and Physics, 2015. **15**(10): p. 5743-5760.
31. Lu, X., et al., *Detection of Fire Smoke Plumes Based on Aerosol Scattering Using VIIRS Data over Global Fire-Prone Regions.* Remote Sensing, 2021. **13**(2).
32. Li, H., et al., *Estimation of leaf water content from hyperspectral data of different plant species by using three new spectral absorption indices.* PLoS One, 2021. **16**(3): p. e0249351.
33. Toulouse, T.R., L.; Campana, A.; Celik, T.; Akhlouf, M. , *Computer vision for wildfire research: An evolving image dataset for processing and analysis.* Fire Safety Journal, 2017. **92**: p. 188-194.
34. Wang, M., et al., *FASDD: An Open-access 100,000-level Flame And Smoke Detection Dataset for Deep Learning in Fire Detection.* Earth System Science Data (Open Acess), 2022. **v3**.
35. Zhao, L., et al., *Investigating the Impact of Using IR Bands on Early Fire Smoke Detection from Landsat Imagery with a Lightweight CNN Model.* Remote Sensing, 2022. **14**(13).

8. Appendix A - List of figures and tables

List of figures

List of tables

9. Appendix B – confidential supplementary parts (access on request)

B.1. Definition of SLOPE1 and SLOPE1 indices

B.2. Key hardware differences between Kanyini and devices used in the emulation system

B.3. Codes for emulation system to be aligned with Kanyini environment

B.4. Emulation experiment - exemplary code

B.5. Commands to convert a trained model on a Windows host computer

B.6. Algorithm for obtaining the band importance using wPCA

B.7. Move function

B.8. Excerpt from the code for data transfer simulation

USER INFORMED INDUSTRY DRIVEN RESEARCH POWERED

SMARTSAT
COOPERATIVE RESEARCH CENTRE

Building Australia's Space Industry

SmartSat CRC Head Office:
Lot Fourteen, Level 2, McEwin Building
North Terrace, Adelaide, SA

info@smartsatcrc.com
smartsatcrc.com