On-board Image Classification Payload for a 3U CubeSat using Machine Learning for On-Orbit Cloud Detection

Mark Angelo C. Purio[1], Timothy Ivan Leong[2], Yasir M. O.Abbas[3], Hoda Awny Elmegharbel[4], Koju Hiraki[5], Mengu Cho[6]

Abstract

CubeSats are giving the opportunity for educational institutes to participate in the space industry, develop new technologies and test out new ideas in outer space. CubeSat missions are developed to perform scientific research and demonstrate new space technologies with relatively cheap cost and limited resources. This category of satellites has many limitations such as the short development time, the power consumption and the limited time and capability of data downlink. Earth Observation from a Low Earth Orbit is one of the most appealing m applications of CubeSats developed by students or non-space faring countries. Investigating new technologies to improve image quality and studying ways to increase acquisition adequacy is very promising. This paper aims to introduce a mission hardware design and machine learning-based algorithm used within an Earth Observation (EO) CubeSat. The case study of this paper is Alainsat-1 project which is a 3U CubeSat developed with the support of IEEE Geo-science and Remote Sensing Society (GRSS) at the National Space Science and Technology Center, UAE. The satellite is planned to be launched by 2022. A low-resolution Commercial off-the-shelf (COTS) camera for EO is developed as a primary mission in this CubeSat. The compatible hardware design and software algorithm proposed is responsible for classifying the images captured by the camera into different categories based on cloud intensity detected in these images before downloading them to the ground station. A microcontroller-based architecture is developed for controlling the mission board; it is responsible for accessing the memory, reading the images, and running the cloud detection algorithm. The cloud detection algorithm is based on a U-net architecture while the algorithm is developed using a Tensor-flow library. This model is trained using a dataset of images taken from the Landsat 8 satellite project. Moreover, the SPARCS cloud assessment dataset is used to evaluate the developed model on a new set of images. The overall accuracy achieved by the model is around 85% in addition to the acceptable performance of the model observed on a set of low-resolution images. The plan is to make the design modular and optimize its performance to be used on-board CubeSats fulfilling the size constraint and overall power consumption limitation of an add-on module to a camera mission.

Keywords

CubeSat, Cloud Detection, Image Classification, U-net architecture, Microcontroller

[1] Corresponding author: Kyushu Institute of Technology, Japan, purio.mark-angelo894@mail.kyutech.jp
[2] Kyushu Institute of Technology, Japan, leong.timothy-ivan483@mail.kyutech.jp
[3] Kyushu Institute of Technology, Japan, yasir.m.o.abbas@gmail.com
[4] Kyushu Institute of Technology, Japan, elmegharbel.hoda827@mail.kyutech.jp
[5] Kyushu Institute of Technology, Japan, hiraki.koju735@mail.kyutech.jp
[6] Kyushu Institute of Technology, Japan, cho.mengu801@mail.kyutech.jp

1. Introduction

The popularity of CubeSat development due to its cheaper cost and ease of production has allowed testing new ideas in space. However, these CubeSats present limitations in terms of power, size, and downlink capacity. To compensate with these limitations, the development of optimized algorithms which uses limited bandwidth and power and are limited in size are an important part of developing these CubeSats' efficiency.

Cloud detection [1], [2] is an important step for the collection of satellite imagery. In most scientific studies in Earth Observation, cloudy images have little to no use due to its contamination of the satellite image[3]. Since this is the case, the time to download such cloud-contaminated images and their storage may be an additional work to satellite mission execution and operation. This is also true for cube satellites which has relatively limited bandwidth and short data downlink time. Due to this, the development of algorithms able to accurately determine clouds in each image is an interesting and useful topic in CubeSats with imaging payloads.

On board classification algorithm could prove to be a useful tool for these scenarios. Being able to select data useful for the satellite's application could optimize data downstream to the ground station as only useful data are downloaded. In the literature, on-board computing in CubeSats have been implemented in different applications. For instance, Thompson et al (2015) did an onboard machine learning classification of images by a CubeSat in Earth orbit [4] while Thompson et al (2015) and Hernández-Gómez et al (2019) present nine processing-intensive algorithms very commonly used for the processing of remote sensing data which can be executed on-board on this platform [5]. Machine-learning based payload for CubeSats were also done by Manning et al (2018) used various Convolutional Neural Networks (CNNs) to identify and characterize newly captured satellite images [6], and Maskey & Cho (2020) provides an innovative approach of combining a novel CubeSat image dataset and a lightweight Convolutional Neural Network architecture for automatically selecting images for downlink on a 1U CubeSat [7]. More so, cloud detection for a CubeSat were also explored and developed by Zhang et al (2018) to reduce memory cost and interference speed by utilizing image-compression strategy and depth-wise separable convolutions [8] and another work of the same author (2019) use a combination a convolutional neural network and wavelet image compression is proposed to explore the possibility of onboard cloud detection [9].

This paper presents a mission hardware design and machine learning-based algorithm used within a CubeSat called Image Classification Unit (ICU). The case study of this paper is Alainsat-1 project which is a 3U CubeSat developed with the support of IEEE Geo-science and Remote Sensing Society (GRSS) at the National Space Science and Technology Center, UAE. The satellite is planned to be launched by 2022. The compatible hardware design and software algorithm proposed is responsible for classifying the images captured by the camera into different categories based on cloud intensity detected in these images before downloading them to the ground station. A microcontroller-based architecture is developed for controlling the mission board; it is responsible for accessing the memory, reading the images, and running the cloud detection algorithm.

2. Materials and Methods

2.1. Data Preparation

2.1.1. Image Selection

To detect cloud in an image, the model is required to be fed with an image and the mask of cloud associated with the image. It also requires a large amount of data so that it can get high accuracy while not overfitting to its training dataset. Freely available images were used and taken from Landsat 8 courtesy of the US Geological Survey (USGS) from which bulk of the images would form the dataset.

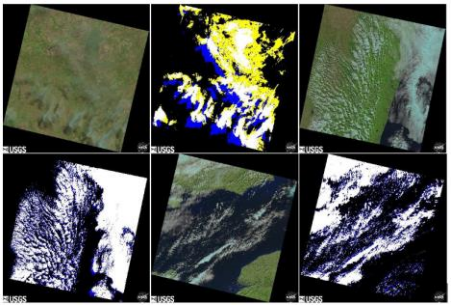

Figure 1. Example of images from Landsat 8 alongside its quality band (Images courtesy of the U.S. Geological Survey)

The model has been trained using only the RGB band of an image as this would probably be what most cube satellite would be able to capture. With this process around 200 images were downloaded from the USGS website.

2.1.2. *Image Pre-processing*

Each image and its respective quality band needed to be pre-processed to use them to train the model. The following workflow was done during this stage:

1. Manual reorientation of images and resizing to 6000 x 6000 pixels.
2. Slicing the images to multiple smaller images of 500 x 500 pixels.
3. Transforming quality bands into black and white to keep cloud position information.
4. Normalizing all coefficients to 0 or 1 before including in the dataset.
5. Resizing the 500 x 500-pixel images to 256 x 256 by doing an average of color over a region of pixels.
6. Feeding the model with the resized images.

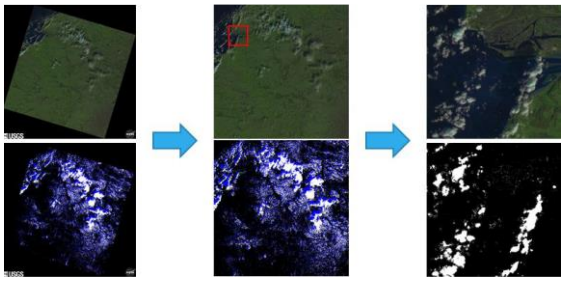

Figure 2. Processing of a Landsat 8 image to an image used in our dataset

The last two processes down samples the Landsat 8 images from 30m resolution to about 300m resolution to fit normal CubeSat images After processing a new dataset consisting of 15,263 images was generated

2.1.3. *Evaluation Dataset Selection*

To properly evaluate the model, the SPARCS cloud assessment dataset [10] was used as a validation dataset. The 80 scenes of the dataset contain all types of photos a cloud detection algorithm might encounter and thus is very useful to evaluate such an algorithm. An evaluation dataset is also necessary to evaluate the model on images it has not been exposed to.

Images from CubeSats such as CubeSats such as Horyu-4, BIRDS 3, Calpoly, and MySat1 were also gathered to evaluate the machine learning model on images of lower resolution. These images could not be used in the dataset as they didn't have any pre-generated cloud mask. Thus, the prediction made by the model on such images will only be empirically evaluated.

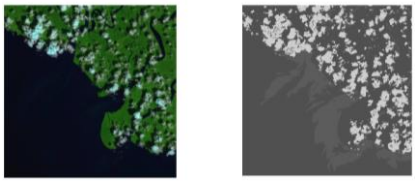

Figure 3. Example of a SPARCS images and their associated cloud mask

2.2. *Algorithm Implementation*

Normal CNN implementation is not enough for our purpose as it tends to extract the features out of an image while losing any spatial information about these features. To output a mask of cloud, this lost spatial information need to be regained. This is done using a U-net architecture designed by Ronneberger et. al. in 2015 [11]. It consists of a contracting path that is basically a normal CNN network and a symmetrical expanding path. The contracting path extract the features from the image, while doing so the tensors height and width get smaller and smaller and only the features are left. As seen in Figure 4, each down sampling divides the height and width of the tensor by 2 while the depth of the tensor increase, i.e., the number of the extracted features increases. To upscale the tensor back to the input tensor size, a 2x2 convolution is used (Up-convolution). Since this up-convolution half the number of features of its input tensor, a concatenate operation is done with the corresponding tensor from the contracting path.

2.3. *Training and Optimization of the Model*

Having built the dataset that will train our model and defined the neural networks architecture used for our model, training and optimizing said model becomes the next step. The first model trained achieved an accuracy of around 85%. To get a more accurate model, multiple parameters could be changed:

- The size of the input image and the augmentation of the images in the training dataset
- The size of the training batch and validation batch of images fed to the model each iteration before updating the weights.
- The number of Epoch

- The number of layers and the number of filters implemented in the neural network
- The dropout parameter
- The optimizer function used in the architecture
- The threshold at which a pixel is deemed to be a cloud

For this purpose, a multitude of model with varying parameters were trained and evaluated based on their accuracy and if they were overfitting or not to their training dataset. The duration of each simulation lasting around 30 minutes, the task was relatively long to accomplish and the influence of each parameter on the accuracy of the model could only be interpreted with a relatively few numbers of points for each parameter.

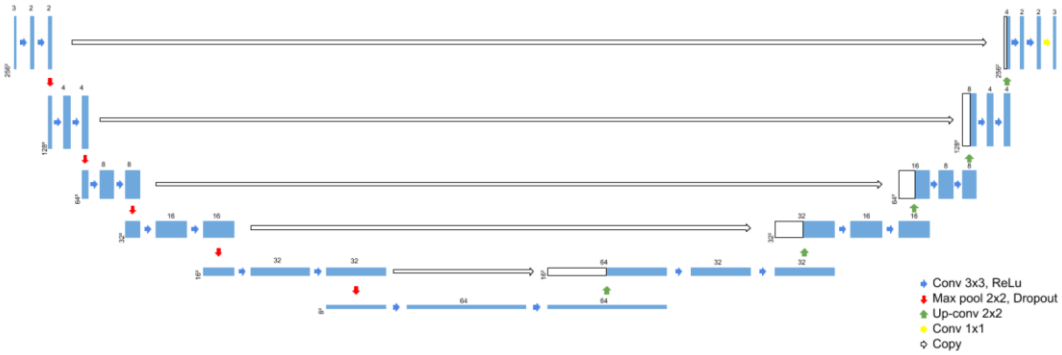

Figure 4. Architecture of a U-net Convolutional neural network, inspired from Ronneberger et al. (2015) [11]

2.4. Microcontroller Implementation

To implement the newly built model inside a microcontroller, several things were done:

1. Quantizing the model to reduce to a size suitable for the microcontroller RAM.
2. Converting the model to C++ code array so it can run in the microcontroller.
3. Write an algorithm that will (a) retrieve an image from the flash memory, (b) convert it to the input tensor format used by the model, (c) run the model to retrieve the resulting cloud mask and (d) save it in a flash memory.

2.5. Performance Evaluation

After training the dataset, the model was evaluated and by calculating its accuracy over the training dataset, the validation dataset, and the evaluation dataset. The model was evaluated based on its confusion matrix, precision, recall, false omission rate, commission rate, overall accuracy and the F1 score. The model was also evaluated on the nanosatellite images to see if it could properly work with lower resolution images.

2.6. Payload Hardware Design

Before the PCB was designed, the model was first tested inside a STM32F746G-DISCO development kit. This development kit offers similar specification as the one the final PCB design so making the model work inside this board will guarantee that it will work inside the other board.

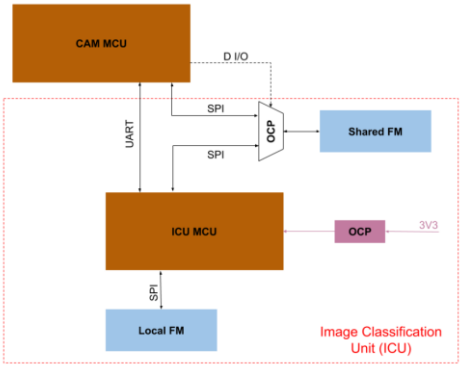

Figure 5. Basic schematic of the ICU Payload hardware

3. Results and Discussion

3.1. Selected Model

In Table 1 the parameters that were set for the best model is shown. Multiple models had higher accuracy than the one presented here, but once they went through the quantization process, their size remained too big to be usable inside the microcontroller. In the end the model size managed to be reduce to 867 Kb.

Table 1. Model Parameters Used

Parameters	Value	Reason
Size of the image	256 x 256	Limit the memory usage while keeping high enough details
Image Augmentation	Yes	Make the model react better to various type of images
Training Batch	15	Update the weights as often as possible
Validation Batch	5	Validation dataset is smaller
Number of Epoch	15	Model reached a plateau around this amount of Epoch
Number of layers	5	Number of layers limited by the microcontroller RAM
Number of starting filters	2	Number of filters limited by the microcontroller RAM
Dropout	0.3	Best results
Loss function	Binary Cross entropy	Best loss function to use in this case
Optimizer function	Adam	Best results
Mask threshold	0.5	Best results

3.2. Performance Evaluation

The chosen model was tested using the different performance metrics and is summarized in Table 2. On the other hand, the confusion matrix for the training, validation and evaluation dataset is shown in Table 3 to provide a better visual representation of the performance metrics. Figure 6 presents sample predictions on training, validation, SPARCS, and CubeSat images.

Table 2. Accuracy Assessment of the model

Method	Training Dataset	Validation Dataset	Evaluation Dataset
Precision	0.7297	0.8246	0.6151
Recall	0.6969	0.7313	0.5373
False Omission Rate	0.1565	0.1529	0.1079
Commission Rate	0.2703	0.1754	0.3849
Overall Accuracy	0.8060	0.8396	0.8452
F1 score	0.7129	0.7751	0.5736

Table 3. Confusion Matrices for training dataset, validation dataset, and evaluation dataset.

Training Dataset		**Predicted Class**	
		Cloud	Non-Cloud
Actual Class	Cloud	0.6969	0.3031
	Non-Cloud	0.1364	0.8636
Validation Dataset		**Predicted Class**	
		Cloud	Non-Cloud
Actual Class	Cloud	0.7313	0.2687
	Non-Cloud	0.0946	0.9054
Evaluation Dataset		**Predicted Class**	
		Cloud	Non-Cloud
Actual Class	Cloud	0.5373	0.4627
	Non-Cloud	0.0808	0.9192

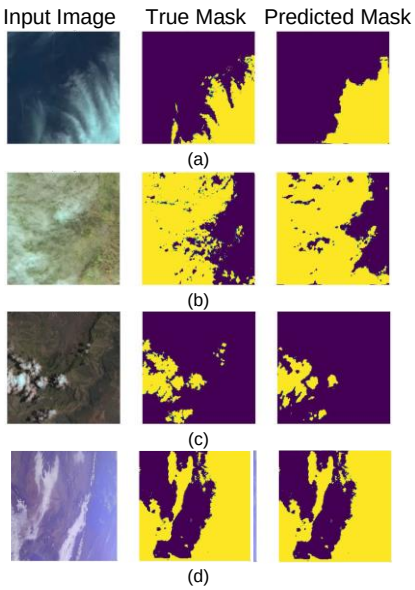

Figure 6. Example predictions for (a) training, (b) validation, (c) SPARCS and (d) CubeSat images

3.3. Payload Design

The payload was design and is currently under manufacturing. Figure 7 shows the PCB design and the 3D render of the board. Such board includes both the image classification unit payload and the camera payload designed by Telkom University.

Figure 7. PCB design and 3D render of the payload (Credit: Telkom University)

Considering the lessons learned in the BIRDS-4 satellite project, a PCB is to be fabricated to implement the software. It consists of a microcontroller, one shared Flash memory with the camera microcontroller and one local flash memory. In addition, it is integrated with the camera payload designed by Telkom University. The final purpose of this system is to have a cheap module that can be implemented on as many CubeSats as possible, so the hardware design was kept simple and replicable. All the chosen components have space heritage already. This will ensure that the new PCB design should be able to work in the harsh environment of space. Both radiation, thermal and vibration test will still be conducted on the new PCB to ensure its resistance to the space environment. One of the main challenges faced by the design is the RAM size, with the cloud detection the amount of stored data in the

RAM and in the Flash is huge because of weights and activations of the model. The size of the Flash in the STM32F7 device is enough but we needed to add the external RAM. To provide hardware acceleration for the JPEG decoder, we are considering using STM32F767 instead of STM32F746. They have pin to pin compatibility.

4. Discussion and Conclusions

A U-net convolutional neural network was designed and trained to be implemented inside a CubeSat payload. For this purpose, satellite images from Landsat 8 were gathered and processed to generate a new dataset consisting of 15,263 images.

After a lot of models were trained with different parameters to obtain the best model possible, several models were selected and quantized to observe the influence of the size of the model's architecture on its final optimized size. The chosen microcontroller's RAM size of 2 Mb limits the selection of high accuracy models which are big in size, but additional RAM is found to be a better improvement.

The final chosen model manages to obtain good result considering the size limitations when it was significantly reduced in size via a quantization method. It has been properly loaded inside an MCU proving that the model will eventually be able to run inference. Since the model doesn't seem to be losing accuracy when exposed to high reflective area and the relative shape and area of the clouds is relatively respected by the model, the results were deemed satisfying for the purpose of this paper. Thus, even if the evaluation score is lower than what normal cloud detection algorithm can achieve, implementation inside a microcontroller will still be conducted and tested.

5. Future Work

The problem of the size versus the accuracy model will have to be one of the main themes of any future work. They will have to work both on the issues the algorithm has with running in the microcontroller and changing the architecture of the model to obtain better results while keeping the model small enough to make it run inside the microcontroller. Integrating the quantization process during the learning step of the neural network could be one method to explore to allow this compromise between size and accuracy.

To date, the payload hardware is still under manufacturing so full implementation results are to be discussed in the future.

Acknowledgements

- The Laboratory of Lean Satellite Enterprises and In-Orbit Experiments of Kyutech for supporting to carry out the project and allowing the use of its laboratory equipment and facilities.
- BIRDS-4 Satellite project and its members, as this project is where the mission idea originated.
- Funding support from IEEE Geo-science and Remote Sensing Society (GRSS) at the National Space Science and Technology Center, UAE through the Alainsat-1 project.

References

[1] J. H. Jeppesen, R. H. Jacobsen, F. Inceoglu, and T. S. Toftegaard, "A cloud detection algorithm for satellite imagery based on deep learning," *Remote Sens. Environ.*, vol. 229, pp. 247–259, Aug. 2019.

[2] T. Bai, D. Li, K. Sun, Y. Chen, and W. Li, "Cloud Detection for High-Resolution Satellite Imagery Using Machine Learning and Multi-Feature Fusion."

[3] K. Vani and G. U. V. L. Priya, "Detection and removal of cloud contamination from satellite images," *https://doi.org/10.1117/12.697138*, vol. 6408, pp. 79–87, Dec. 2006.

[4] D. R. Thompson *et al.*, "Onboard machine learning classification of images by a cubesat in Earth orbit," *AI Matters*, vol. 1, no. 4, pp. 38–40, Jun. 2015.

[5] J. Hernández-Gómez *et al.*, "Conceptual low-cost on-board high performance computing in CubeSat nanosatellites for pattern recognition in Earth's remote sensing," Kalpa Publications in Computing, 2019.

[6] J. Manning *et al.*, "Machine-Learning Space Applications on SmallSat Platforms with TensorFlow."

[7] A. Maskey and M. Cho, "CubeSatNet: Ultralight Convolutional Neural Network designed for on-orbit binary image classification on a 1U CubeSat," *Eng. Appl. Artif. Intell.*, vol. 96, p. 103952, Nov. 2020.

[8] Z. Zhang, G. Xu, and J. Song, "CubeSat cloud detection based on JPEG2000 compression and deep learning," *Res. Artic. Adv. Mech. Eng.*, vol. 10, no. 10, pp. 1–10, 2018.

[9] Z. Zhang, A. Iwasaki, G. Xu, and J. Song, "Cloud detection on small satellites based on lightweight U-net and image compression," *J. Appl. Remote Sens.*, vol. 13, no. 02, p. 1, 2019.

[10] M. J. Hughes and R. Kennedy, "High-Quality Cloud Masking of Landsat 8 Imagery Using Convolutional Neural Networks," *Remote Sens. 2019, Vol. 11, Page 2591*, vol. 11, no. 21, p. 2591, Nov. 2019.

[11] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9351, pp. 234–241, May 2015.