On-orbit rule-based and deep learning image segmentation strategies

Shreeyam Kacker*, Alex Meredith*, Violet Felt[†], Joe Kusters*, Hannah Tomio*, Kerri Cahoy [‡]
Massachusetts Institute of Technology, Cambridge, MA 02139

We discuss image segmentation algorithms and additional space considerations for BeaverCube-2, a project under development between the MIT Space Telecommunications, Astronomy, Radiation (STAR) Lab and the Northrop Grumman Corporation that aims to demonstrate the use of an Artificial Intelligence (AI) Computational Accelerator System-on-a-Chip (SoC) on a 3U CubeSat in Low-Earth Orbit (LEO). The processing power afforded by the SoC will allow the use of modern artificial intelligence techniques as part of an Earth observation mission to obtain and process visible and infrared imagery of coastal features.

We focus on three algorithms used for cloud segmentation in satellite imagery. These are a luminosity-thresholding method, a random forest method, and an autoencoder-based deep learning method. Our luminosity thresholding method classifies each pixel based on its luminosity and achieved 84% accuracy using 2 MB of memory. Our random forest method contextualizes pixels within a 3×3 kernel and classifies them based on the luminosity of each pixel in the kernel – it achieved 90% accuracy, with a memory usage of 700 MB. Finally, our U-Net-based deep learning method achieved 92% accuracy with 1500 MB memory usage, demonstrating modest gains over the two simpler methods, with higher accuracy in snow scenes.

I. Introduction

Image segmentation and classification are usually performed on the ground on downlinked remote sensing data. However, this requires downlinking all data over potentially constricted bandwidths and short passes over ground stations. Ranking images on-orbit and downlinking only high-quality images can therefore increase useful data throughput. In this work, we focus on the problem of segmenting clouds in satellite imagery with limited computational resources for the BeaverCube-2 mission.

BeaverCube-2 is an Earth observation CubeSat mission jointly developed by the MIT Space Telecommunications, Astronomy, and Radiation Laboratory (STAR Lab) in collaboration with and with the support of the Northrop Grumman Corporation (NGC). BeaverCube-2 follows BeaverCube-1 [1], imaging oceanographic features of interest with visible spectrum and infrared spectrum cameras. BeaverCube-2 seeks to identify, on-orbit, image features of interest, such as coastlines or sea surface fronts, using a modern Artificial Intelligence (AI) Computational Accelerator System on Chip (SoC). This SoC will enable BeaverCube-2 to use modern computer vision techniques to perform accurate identification of desirable science features, such as coastlines or sea surface fronts, and undesirable features, such as clouds.

BeaverCube-2 intends to image coastal regions around Cape Hatteras in North Carolina and identify oceanographic features of interest on-orbit, prioritizing desirable images for downlink. Its concept of operations is shown in Figure 1. We are developing a model that allows BeaverCube-2 to identify temperature and chlorophyll-a ocean fronts. The quantity and quality of fronts in each image will determine the image's downlink priority. This model only performs well on images with <5% cloud coverage. In images with greater cloud coverage, the locations (or existence) of ocean fronts in the image can be obscured, resulting in inaccurate downlink priorities. Only selecting and processing high-quality images with low cloud coverage as part of the computer vision pipeline shown in Figure 2 lowers on-orbit power requirements and improves useful data throughput, lowering the number of ground station overpasses required. Executing computationally intensive image segmentation algorithms for cloud identification on-orbit will also help BeaverCube-2 meet its primary technology goal, demonstrating the use of the AI Accelerator SoC on-orbit.

*Graduate Student, Aeronautics and Astronautics
[†]Graduate Student, Electrical Engineering and Computer Science
[‡]Associate Professor, Aeronautics and Astronautics

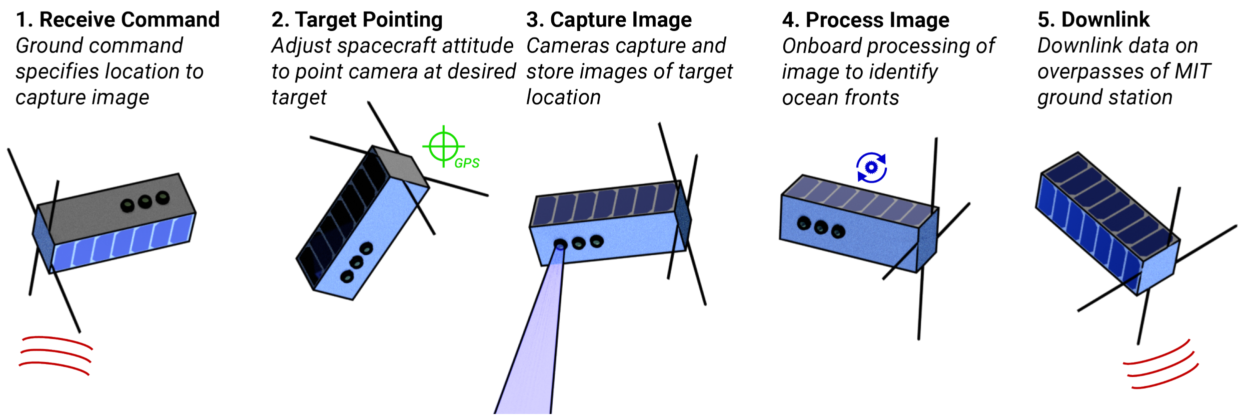

1. Receive Command
Ground command specifies location to capture image
2. Target Pointing
Adjust spacecraft attitude to point camera at desired target
GPS
3. Capture Image
Cameras capture and store images of target location
4. Process Image
Onboard processing of image to identify ocean fronts
5. Downlink
Downlink data on overpasses of MIT ground station

Fig. 1 Concept of operations for the BeaverCube-2 mission

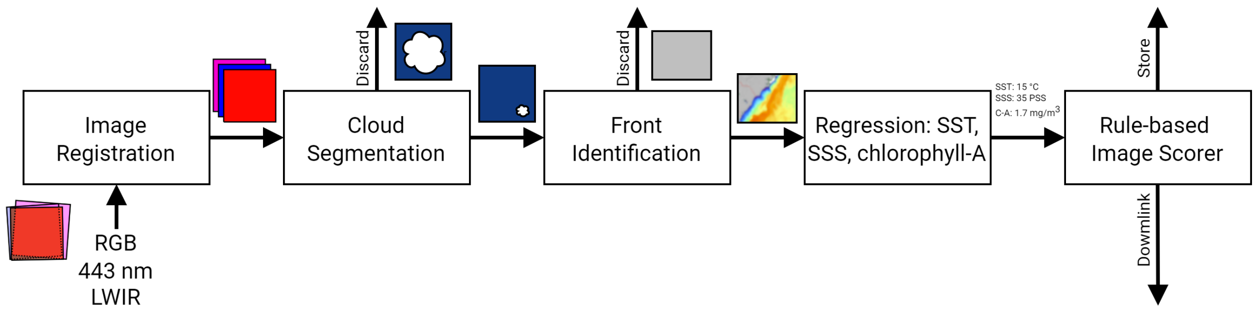


Fig. 2 Computer vision pipeline to obtain sea surface temperature (SST), salinity (SSS), and chlorophyll-A concentration

II. Background

State-of-the-art approaches to cloud detection in satellite imagery use several different methods. These vary from rule-based cloud detection schemes like Fmask [2] and s2cloudless [3], to the machine learning methods like random decision forest, Bayesian thresholding, and salience-based novelty detection used on the EO-1 spacecraft [4], to deep learning methods for cloud detection that have yet to be demonstrated on-orbit. This work covers the usage of a luminosity-based detection method, a random forest (RF) based detection method, and a deep learning detection method implemented as a U-Net [5].

A. Classical Cloud Segmentation

A classical method is long lead-time median filtering for removing clouds from images [6]. This method compares spatially correlated images taken at different times and takes the median value of each pixel over many spatially correlated images to remove clouds, which vary dynamically from image to image. With sufficient numbers of images, this method almost entirely eliminates clouds from images, as shown in Figure 3. Long lead-time median filtering is well-suited to on-the-ground image processing for satellite imagery from missions with many ground passes over the same areas, where precise spatial data is available. However, it is not well suited to on-orbit processing due to storage and lead-time constraints.

B. Rule-based Methods

Rule-based image segmentation provides shorter lead time segmentation based on features present in single images. Rule-based methods use computationally simple rules for segmentation, often comparing data from different sensor bands to human-selected threshold values. Another rule-based cloud detection approach involves using random forests, which train on image data to develop decision trees that maximize information gain at each split in the tree. Although

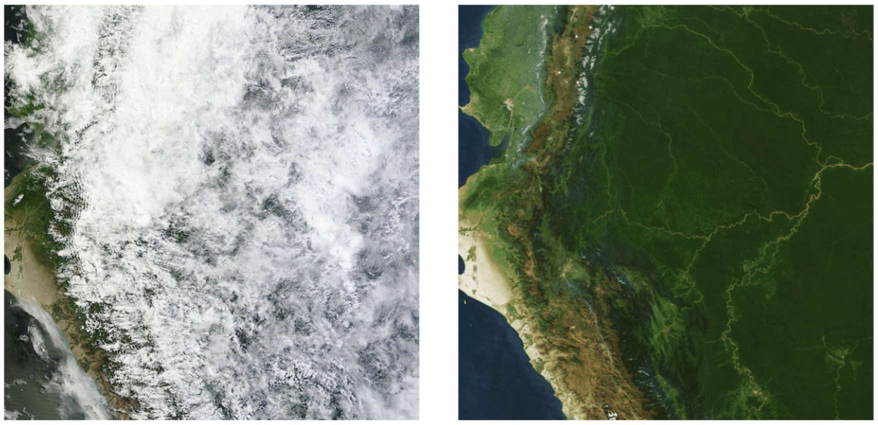


Fig. 3 MODIS median filtered image, showing one of many images on the left and the median output on the right [7]

more sophisticated than basic human-selected thresholds, random forests classify pixels based on a series of preset rules, and are considered a rule-based method.

The Earth Observing 1 (EO-1) spacecraft tested two rule-based cloud detection methods on-orbit: Bayesian thresholding (BT), which classified pixels as cloud or not cloud using simple luminosity thresholds in two visible bands and one infrared band, and a random forest that classified pixels based on the properties of a surrounding 5×5 pixel window in three visible bands [4]. Both classifiers were relatively slow to run on-orbit; the random forest classifier took up to 52 minutes to run [4]. Although the number of on-orbit classifications was limited, the random forest method was able to classify 19 images on EO-1, with two images falsely tagged as containing clouds when there were none and two images falsely tagged as cloudless when clouds were present. The remaining 15 images classified correctly [4]. During ground testing, both classifiers used on EO-1 showed relatively high sensitivity but poor specificity, a common pitfall for rule-based cloud detection algorithms [4].

Although EO-1 used only three bands in the visible and infrared spectrum, many rule-based cloud detection approaches use several bands. For example, some satellites equipped with microwave radiometers use microwave-band measurements to provide additional data for identifying clouds and precipitation [8]. More commonly, on imaging satellites like Landsat and Sentinel-2 with bands in the visible and infrared, rule-based methods take advantage of data from most or all bands to improve accuracy. In particular, thermal infrared (TIRS) data helps augment data from optical bands when identifying clouds [2].

Fmask, a well-known rule-based cloud detection method originally developed for use with Landsat data, extensively uses many bands on Landsat for cloud identification. Fmask uses several rules with hand-selected thresholding to identify potential cloud pixels and to separate water and land pixels, then computes cloud probabilities over land and water pixels [2]. In its original formulation, Fmask was designed to work for Landsats 4-7, which only had seven bands, which are roughly equivalent to bands 2-7 and 10 on the current Landsat 8 and 9 satellites [9]. Fmask extensively used thermal data from wavelengths between 10400 and 12500 nm to differentiate between cloud and non-cloud pixels [2]. Later, Fmask was extended to use the new "cirrus" band on Landsat 8 to aid with cirrus cloud identification and work with Sentinel-2 data [10]. However, due to Fmask's dependence on thermal data and Sentinel-2's lack of a TIRS band, Fmask generally performs better on Landsat 8 data than on Sentinel-2 data [10].

Another rule-based scheme, s2cloudless, was developed specifically for Sentinel-2, and uses gradient-boosted random forests [3]. S2cloudless uses all of Sentinel-2's bands except for 3, 6, and 7 and uses features comprised of raw data from each band, pairwise differences between bands, pairwise ratios between bands, and data averaged over pairs of bands [3]. S2cloudless outperforms Fmask on Sentinel-2 imagery by approximately ten percentage points [3].

One major drawback seen across rule-based methods is poor specificity – many rule-based methods, including Fmask and s2cloudless, tend to misidentify bright ground pixels (like snow) as clouds [3]. One potential explanation for this is that rule-based methods generally identify each pixel individually and cannot differentiate between snow and cloud pixels based on context or texture. Although Wagstaff et al. used 5×5 pixel kernels for random forest classification on EO-1, providing some image context, their random forest classifier still shows relatively poor specificity, perhaps due to its small kernel size [4].

Table 1 Payload Specifications [13–17]

Type	Model	Spectral Range	Pixel Size [μm]	Lens Focal Length [mm]	GSD [m]
Color Visible Camera	Matrix Vision Gmbh mvBlueFOX-IGC 200wC	390 - 700 nm	6	16	150
Narrowband Visible Camera	Matrix Vision Gmbh mvBlueFOX-IGC 200wG	437 - 447 nm	6	16	150
LWIR Camera	FLIR Boson 640 Radiometric	7.5 - 14 μm	6	18	267

C. Deep Learning Methods

Deep learning provides another set of methods for on-orbit cloud identification but has yet to be tested on orbit. Li, et al. proposed using multi-scale convolutional feature fusion (MSCFF), similar to a conventional U-Net [5] for cloud identification [11]. This method outperforms rule-based methods like Fmask, but also tends to misidentify snow pixels as clouds and has difficulty identifying cloud shadows [11]. There is a clear gap in the literature regarding on-orbit cloud detection using deep learning. There is clear room for improvement in designing a deep learning method for cloud detection that can accurately identify snow and cloud shadows, with a goal of doing this on-board a satellite to help with prioritizing science data for downlink. This paper seeks to develop such a deep learning approach, improve on previous work, and compare this approach in terms of accuracy and computational cost to the performance of simpler, rule-based methods.

III. Payload Hardware

A. Cameras and Optics

The BeaverCube-2 imaging payload will obtain coastal and ocean imagery to serve as inputs for the AI-based image processing demonstration and support scientific study of ocean fronts. Based on commercial-off-the-shelf (COTS) components, the payload consists of two visible and one long-wave infrared (LWIR) camera. Of the visible spectrum cameras, one features a detector equipped with a Bayer filter to produce color images, while the other targets the 443 nm band through an external narrowband filter. This targeted ocean science band is intended for observing concentrations of chlorophyll-a, which can be used to estimate phytoplankton biomass and the distribution of algal blooms and has implications for the detection of ocean fronts [12]. In addition, the LWIR camera will supplement these visible spectrum images with thermal imagery to facilitate object identification and enable observations of sea surface temperature (SST), which will also aid in the detection of ocean fronts.

Though intended for terrestrial machine vision applications, the COTS cameras have low size, weight, and power (SWAP), meeting the constraints of the 3U CubeSat platform. The visible spectrum cameras are both mvBlueFOX-IGC 200w models, in color and monochrome [13]. The latter will be equipped with an Edmund Optics 442 nm center wavelength (CWL) 10 nm bandpass filter [14]. Both visible cameras will also be equipped with Tamron M118FM16 16 mm focal length lenses [15]. The LWIR camera is a FLIR Boson 640 Radiometric version with an 18 mm 24°field-of-view (FOV) lens [16, 17]. The use of COTS components avoids the complicated and costly challenges of developing an entire Earth observation payload, which involves custom optics, electronics, or thermal management. The components of the payload, including the cameras, filters, and lenses, were selected after several trade studies, and the payload on-orbit performance has been simulated using a radiometric link analysis. Key payload specifications are summarized in Table 1, including the ground sample distance (GSD) calculated for a 400 km orbit.

B. Electronics

In order to use the AI Accelerator SoC to perform the algorithms detailed in this paper, memory and power requirements need to be met. Specifically, 4 Gigabits (Gb) of nonvolatile memory, 16 Gigabytes (GB) of volatile memory, and a slew of power rails need to be provided along with accompanying thermal considerations.

1. Nonvolatile Memory

Two options are available to meet the 4 Gb requirement on the nonvolatile memory, NOR and NAND Flash memory chips. Radiation reliability concerns are detailed in papers such as [18], but the main takeaway is that NOR Flash is more reliable owing primarily to the lack of bad blocks present in the chips and singular nature of the memory cells reducing failure modes. This is balanced against the storage density achievable with NAND Flash, but chipmakers such as Micron produce NOR Flash with up to 2 Gb of memory, and the AI Accelerator SoC is capable of "stacking" two of these chips together, meeting the 4 Gb requirement.

2. Volatile Memory

To meet the 16 GB requirement on the volatile memory, there are variety of options available. The AI Accelerator SoC supports several protocols to communication with volatile memory chips, such as LPDDR4. The main trades are on power draw, read/write speed, and reliability. Only chips which supports error correction codes are looked into for the reliability concerns.

3. Power Electronics

The AI Accelerator SoC requires several different voltage rails to operate, as the internals of the SoC are broken up into several power domains. This has advantages as it allows finer control over what is on within the SoC, and for the CubeSat mission allows us to keep the SoC on but in a low power state when not processing images. While there are several rails on the SoC, the main power draw is from the Accelerator domain of the SoC, accounting for up to 80% of the chip's power draw, with 15% from the DDR controller needed to process the images. This is in a 25W power draw scenario, and when not using the Accelerator domain, the SoC can draw under 1 W of power. To accommodate the large power draw of the Accelerator domain, this domain is powered from a 6-Phase Buck Converter to dissipate the heat better. This also reduces inductor and capacitor requirements while only requiring a small amount more space. The other domains of the SoC are all powered by a Power Management Integrated Circuit (PMIC), which controls and drives several single-phase buck converters.

IV. Image Analysis Approach

A. Dataset Creation

1. Remote Sensing Data

While many datasets exist for RGB or RGB + near infrared (NIR) cloud segmentation implementations [11, 19], LWIR has not been used in many deep learning based cloud segmentation implementations and the authors could not find a dataset. Therefore, we opted to create our dataset with bands chosen to maximize utility for our mission [*].

Remote sensing data was used from Landsat 8, as it has both the Operational Land Imager (OLI) and Thermal Infrared Sensor (TIRS) bands available to use, which most closely match the imaging systems on BeaverCube-2. Although MODIS also has these bands available, combinations of bands with more discontinuities in spectral sensitivity would be used. A complete list of bands available in Landsat 8 remote sensing imagery is shown in Table 2

Landsat 8 Level 1 data was used with no postprocessing performed, using raw images from sensor calibration. We used bands 2, 3, 4, 10, and 11 as they were the most relevant to our mission, while also downloading scenes in band 5 to compare our cloud segmentation algorithm to other implementations [19].

We downloaded patches of size 256×256 as 24-bit TIFF files at a resolution of 90 m/px from Google Earth Engine with bands 2, 3, 4, 5, 10, and 11 being used. Bands 2, 3, and 4 were combined into a true-color RGB image, while band 5 was saved as a grayscale image, and bands 10 and 11 were superposed and saved as a grayscale image. The cloud reference mask was also downloaded from the Quality Assessment (QA) band and saved as an image. Additionally, scenes were classified by type as arid, coastline, forest, ocean, plains, snow, and urban. This builds on work of categories on other datasets [11], and allows us to see where our model performs poorly and formulate improvements. Scenes were balanced across all categories to not introduce any bias towards one particular scene into models during training. One challenge is that Landsat does not image much ocean, which is mitigated by taking ocean samples around small islands surrounded by deep ocean. The most relevant scene type to BeaverCube-2's mission is coastline scenes, as we wish to identify chlorohpyll-a concentrations, which are most prominent along coasts.

[*] Available at `https://github.com/MIT-STARLab/LWIR-Cloud-Dataset`

The combination of these bands covers a similar spectral range to the imaging frontend on BeaverCube-2, although with a slightly smaller LWIR band with a discontinuity in the center.

Table 2 Landsat 8-9 OLI and TIRS bands, spectral ranges, and ground sampling distance (GSD), with bands in bold included in dataset [20]

Band	Name	Spectral Range [μm]	GSD [m]
1	Coastal aerosol	0.43-0.45	30
2	**Blue**	**0.45-0.51**	**30**
3	**Green**	**0.53-0.59**	**30**
4	**Red**	**0.64-0.67**	**30**
5	**Near Infrared (NIR)**	**0.85-0.88**	**30**
6	SWIR 1	1.57-1.65	30
7	SWIR 2	2.11-2.29	30
8	Panchromatic	0.50-0.68	15
9	Cirrus	1.36-1.38	30
10	**Thermal Infrared (TIRS) 1**	**10.6-11.19**	**100**
11	**Thermal Infrared (TIRS) 2**	**11.50-12.51**	**100**

2. Mask Limitations

The cloud flag on Landsat 8's QA bands are accurate, but can often show false positives in specific scene types. Most commonly, snow, coastlines, and concrete building roofs can trigger false positives. In addition, for thin clouds the exact boundary definitions can be considered subjective, and can be overexpanded. These issues were manually resolved in our dataset, as otherwise our models would produce the same false positives. This dataset was reviewed again to make sure that the human-labelled edits were correct.

3. Data Augmentation

An 80/20 train/test split was created from the full dataset of 840 patches, with balanced images across scene types. Images in the dataset were initially sampled at 90 m GSD and resampled to 150 m GSD once loaded into the model. Additionally, images are randomly flipped horizontally or vertically, with 50% probability each. This increases the effective sample space of the dataset and helps reduce bias.

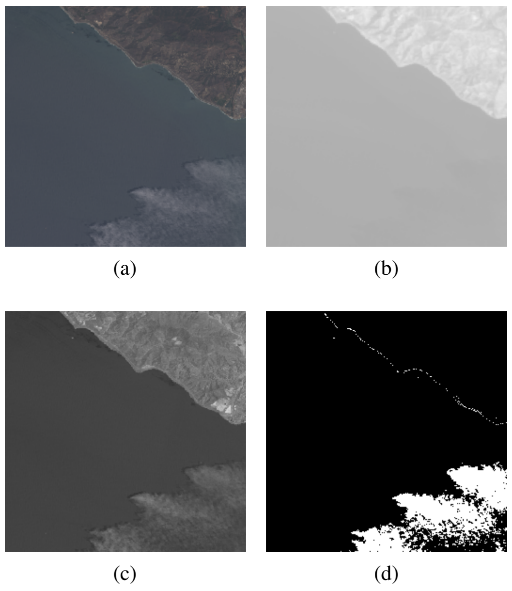


Fig. 4 QA band cloud mask fooled by coastline: (a) RGB; (b) LWIR; (c) NIR; (d) Cloud mask.

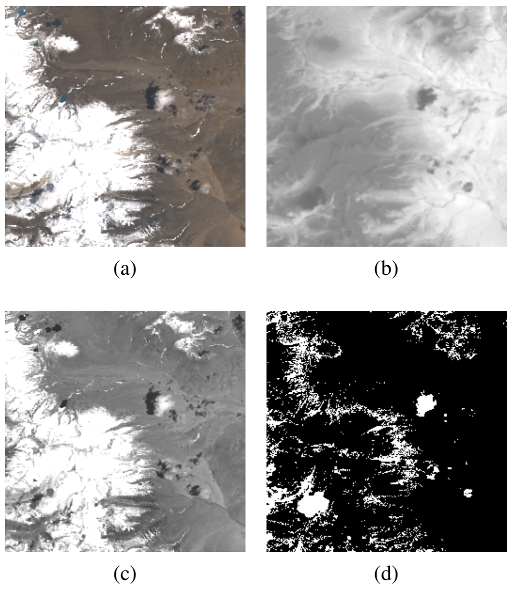


Fig. 5 QA band cloud mask fooled by snow: (a) RGB; (b) LWIR; (c) NIR; (d) Cloud mask.

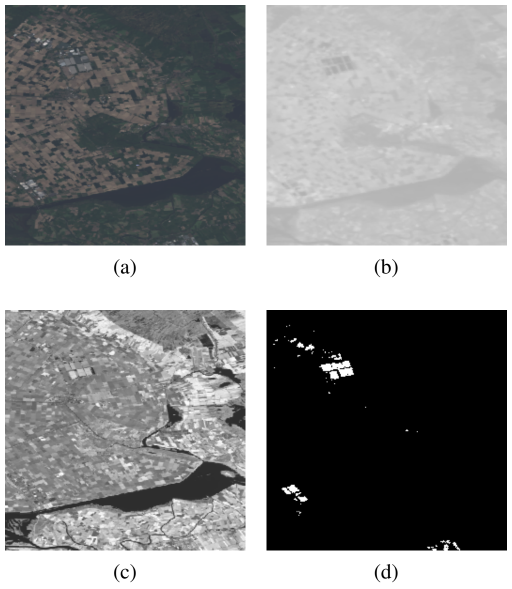


Fig. 6 QA band cloud mask fooled by building tops: (a) RGB; (b) LWIR; (c) NIR; (d) Cloud mask

B. Luminosity Thresholding

Luminosity thresholding is a simple cloud detection method that classifies each pixel based on its luminosity in the red, blue, green, and LWIR channels. Each channel has a cloud luminosity threshold, and if a pixel's luminosity exceeds the cloud luminosity threshold in the blue channel, is below the threshold in the the LWIR channel, or exceeds the threshold in both the red and green channels, it is classified as a cloud pixel. Otherwise, it is classified as a noncloud pixel.

Luminosity thresholding can be considered to be a highly specialized case of the random forest method discussed in section IV.C, where three depth-2 trees vote on whether a pixel is classified as cloud or noncloud, based on red, green, blue, and LWIR luminosity thresholds. The first tree will vote “noncloud" if the pixel luminosity is below the threshold in the blue channel. The second tree will vote “noncloud" if the pixel luminosity is above the threshold in the LWIR channel. The final tree votes “noncloud" only if the pixel luminosity is below the threshold in both the red and green channels. Two votes are required for a pixel to be classified as “noncloud" – otherwise, it will be classified as “cloud". All thresholds are trained individually to minimize false classifications on the training data for a particular channel. Minimizing false classifications also minimizes Gini impurity as given in Equation 1.

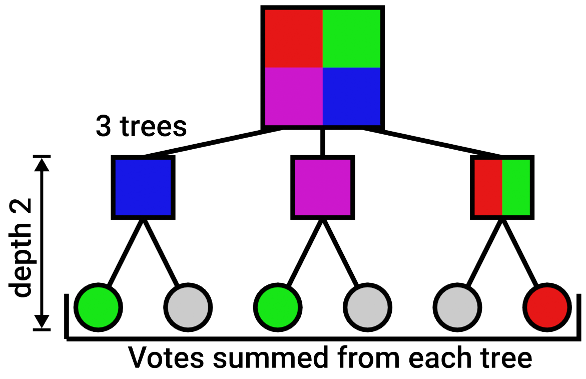


Fig. 7 Luminosity thresholding architecture showing R, G, B, trees voting based on thresholds in 3-tree random forest

Luminosity thresholding is most applicable to identifying targets with a consistent, known luminosity that highly contrasts the background luminosity. Clouds tend to be bright at visible-spectrum wavelengths, particularly at the shorter blue and purple wavelengths. Clouds also tend to be cooler than Earth’s surface, and appear dimmer at LWIR wavelengths, which capture this thermal difference. Luminosity thresholding using visible-spectrum and LWIR-spectrum imagery is well-suited to cloud detection, and is a good candidate for on-orbit cloud detection due to its computational simplicity.

Unlike the other two methods we tested, luminosity thresholding classifies each pixel individually without looking at neighboring pixels. This makes luminosity thresholding fast to train and fast to classify images, but means that luminosity thresholding loses the context of an image and is susceptible to misclassifying aberrant noisy pixels or atypically luminous pixels like cloud shadows or bright coastlines. A diagram of this approach is shown in Figure 7.

C. Random Forest

Random forest (RF) classification is an ensemble learning method that combines n random decision trees trained on different subsamples of data into a forest that typically outperforms any individual classifer [21]. Each random decision tree of depth d classifies a data point by making a series of d decisions based on the data point and its associated features. Then, all n classifications from all n trees are pooled, and the ultimate classification is chosen by plurality vote [21].

RF classifiers are well-suited to on-orbit cloud segmentation because they can classify all pixels in an image relatively quickly, even with significant hardware constraints [4]. Additionally, RF classifiers can be trained on the ground and uplinked to spacecraft, avoiding computationally intensive training on-orbit, provided the feature set remains the same [4]. RF classifiers are also trivially parallelizable due to the same kernel being reused.

1. Random Decision Trees & Impurity

When considering new splits during training, random decision trees seek to minimize a criterion called impurity, which is often considered to be the opposite of information in information theory [22]. The most common impurity functions are based on Shannon entropy and the Gini index [22]. We chose to use Gini impurity, which is quicker to compute because it doesn’t use log functions. Gini impurity is given by the following equation, and represents the likelihood of misclassification of a random data point at a certain node, if the data point were classified randomly [22]:

$$i_g(x) = \sum_{k=1}^{N} p(x)(1 - p(x)) \tag{1}$$

During training, the random decision trees in RF each start with a single node. Then, based on its unique subsample of the training set, each tree finds the optimal way to split the node, maximizing impurity decrease. Each tree then continues to find new splits, building out its nodes while maximizing the decrease in impurity for each split, finally stopping at leaf nodes when maximum tree depth is reached or the minimum number of samples per node is reached [22].

2. Architecture

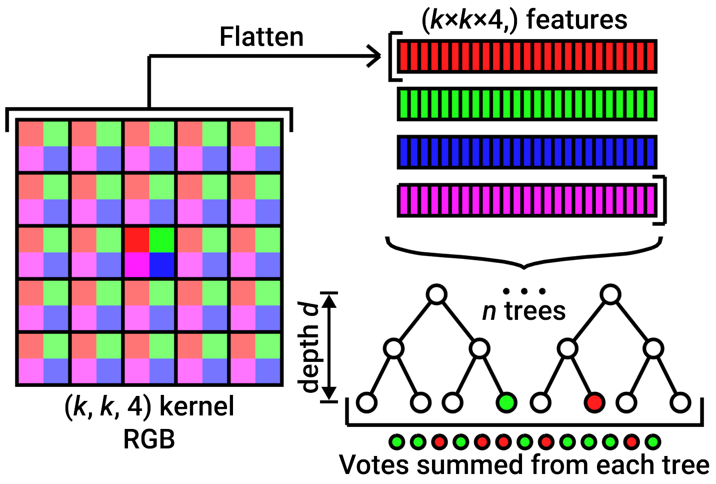


Fig. 8 Random forest architecture, showing classification of a single pixel using a $k \times k$ kernel and a forest of n trees of depth d

We use a kernel-based approach to define features for our random forest, allowing the RF to classify each pixel in context. As such, our RF classifies each pixel based on 36 features: the luminosities in the red, green, blue, and infrared channels for a 3×3 kernel centered on the pixel. A diagram of this is shown in Figure 8. Our cloud classification problem only includes two classes, so final classification is chosen by a majority vote of $> n/2$ trees in a size-n random forest, or ≥ 13 trees in our size-25 random forest. We use Gini impurity as our criterion for finding splits in trees, due to its fast computation time.

In general, RF classifiers show decreases in bias with increases in tree depth d, but also show increasing individual tree variance with increase in depth d [23]. RF variance depends on both individual tree variance and correlation between trees, and may monotonically increase with depth d or follow a U-shaped curve, with high variance at very low or high d, depending on the number of features and size of the training dataset [23]. Choosing an intermediate depth between 1 and the number of features allows a RF classifier to use all useful information in the feature set while avoiding overfitting [23]. We selected a depth-25 random forest for our set of 36 features.

D. U-Net Based Implementation

1. Architecture

The architecture we select is based on a U-Net [5]. U-Nets are similar to autoencoders in that they have have two parts: first, an encoder section creates an abstract lower-dimensional representation on the input. Then, a decoder section uses this abstract lower-dimensional input to construct an output image. However, U-Nets have additional cross-connections, which allow them to produce finer boundaries for segmentation, and also allows them to train faster [24]. In addition, max-pooling operations can cause aliasing and harm the translational equivariance features that convolutional neural networks (CNNs) depend on. Instead, we use an anti-aliased architecture to preserve translational equivariance [25]. The full architecture is shown in Figure 9, with the sizes of each tensor, operations, and cross-connections shown.

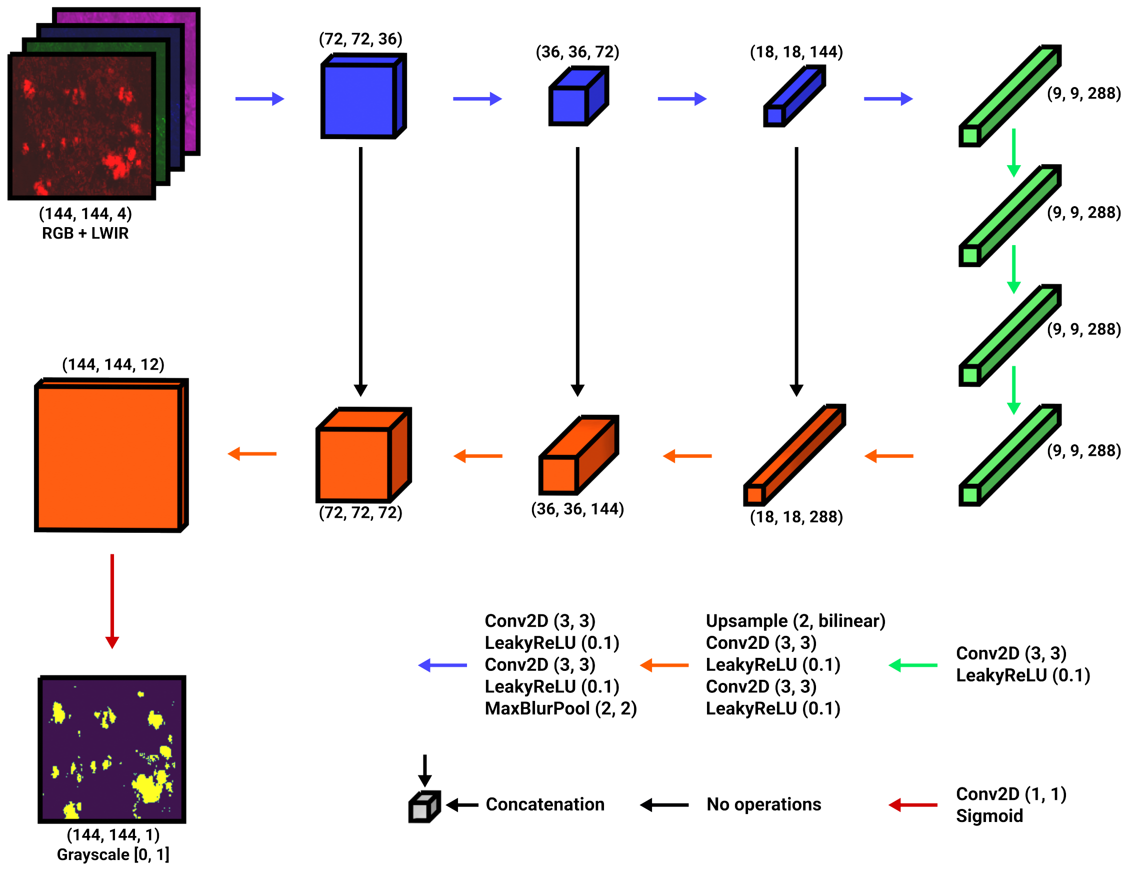


Fig. 9 U-Net based architecture overview

2. Testing scheme and loss function

In the case of image segmentation, Cross Entropy Loss, Jaccard loss, Mean Squared Error (MSE), F1 score, and Dice coefficient are all viable loss functions. As cloud segmentation is a special case of binary segmentation, multi dimensional inputs to a loss function are not required. We chose to use Mean Squared Error (MSE) loss as it is the most simple to use and then be easily extended to provide metrics such as RMS accuracy. Other loss functions could provide additional features that may be desirable depending on the target application, such as penalizing false positive more than false negatives. MSE loss can be expressed as:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2}$$

where y_i is the model output for a particular pixel, $\hat{y}_i$ is the desired output for a particular pixel, and n is the number of pixels an an image.

V. Results

A. Metrics

Each segmentation method outputs a probability density function through which a threshold must then be chosen to output a binary decision. By varying this threshold, the true positivity rate and false positivity rate also vary. The two quantities plotted against each other produce what is known as a receiver operating characteristic curve, representing the family of classifiers available to use depending on where you set your binary threshold. A chance-based classifier can be represented by the $y = x$ curve, whereas the ideal classifier manifests as a step function on an ROC curve. Classifiers below the reference line do not exist, as inversions put flip them around the reference.

Confusion matrices are shown for the particular case of thresholding the output of each segmentation system at 0.5, showing an unbiased estimate of specificity and sensitivity. The ideal classifier has a confusion matrix with probability 1.0 throughout on the leading diagonal, and 0.0 elsewhere.

$$\text{Accuracy} = \frac{\Sigma\,[\text{True Positives} + \text{True Negatives}]}{\Sigma\,[\text{True Positives} + \text{True Negatives} + \text{False Positives} + \text{False Negatives}]} \tag{3}$$

$$\text{Sensitivity} = \text{Recall} = \frac{\Sigma\,\text{True Positives}}{\Sigma\,[\text{True Positives} + \text{False Negatives}]} \tag{4}$$

$$\text{Specificity} = \frac{\Sigma\,\text{True Negatives}}{\Sigma\,[\text{True Negatives} + \text{False Positives}]} \tag{5}$$

$$\text{Precision} = \frac{\Sigma\,\text{True Positives}}{\Sigma\,[\text{True Positives} + \text{False Positives}]} \tag{6}$$

$$\text{F}_1\ \text{Score} = \frac{\Sigma\,\text{True Positives}}{\Sigma[\text{True Positives} + \frac{1}{2}[\text{ False Positives} + \text{False Negatives}]]} \tag{7}$$

All metrics are evaluated on binary-decisions over each pixel for a particular threshold. All of our algorithms perform binary cloud/not cloud classification, so sensitivity and recall are equivalent.

B. Luminosity Thresholding

We selected luminosity thresholds for each channel to maximize correct classifications, using the objective function given in Equation 8. For the red, green, and blue channels, for a given threshold t, we considered true positives to occur when a cloud pixel p_c in the training set had luminosity $\geq t$, and true negatives to occur when a noncloud pixel p_{nc} in the training set had luminosity $< t$, because clouds are reflective and thus have high luminosity in visible-spectrum imagery. For the LWIR channel, we considered the opposite: true positives occurred when a cloud pixel p_c in the training set had luminosity $< t$, and true negatives occurred when noncloud pixels p_{nc} had luminosity $\geq t$. This is because clouds are generally colder than Earth's surface, and so have lower luminosity than noncloud pixels in LWIR imagery.

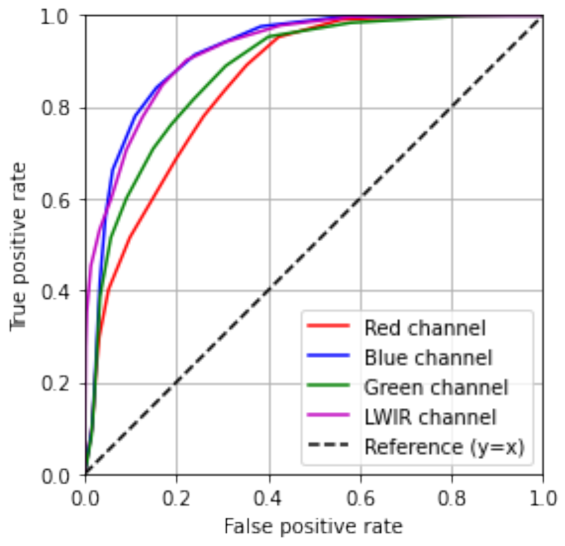
$$\underset{t}{\text{argmax}}\, f(t) = \Sigma[\text{True Positives}(t) + \text{ True Negatives}(t)] \tag{8}$$

In addition to individually training optimal thresholds for each channel, we plot the false positive rate against the true positive rate for different thresholds for each channel, as shown in Figure 10.

As seen in Figure 10, the blue and LWIR channels carry the most information useful for identifying clouds, as these channels are both shifted slightly further to the upper left corner, demonstrating a better true positivity rate than the red and green channels with false positivity rate held constant. We opted to weight the blue and LWIR channels more heavily than the red and green channels for luminosity thresholding: in our luminosity thresholding algorithm, the red and green channels share a 'vote', but the blue and LWIR channels each have their own vote; a pixel needs two votes to be classified as noncloud, or a single vote to be classified as cloud.

Despite its simplicity, luminosity thresholding achieves 84% accuracy, 85% specificity, and 82% sensitivity on average over all scene types, as shown in Figure 11 and Table 3. However, luminosity thresholding performs poorly on arid, coastline, snow, and ocean scenes. In arid, coastline, and snow scene types, luminosity thresholding has low specificity and low precision because it frequently misclassifies visually bright noncloud desert, coastline, and snow pixels as cloud pixels. In particular, snow pixels tend to be bright in visible imagery, but are also cold, and so may appear dim in LWIR imagery. Because luminosity thresholding looks for visually bright and cold pixels, snow pixels are often misclassified as clouds.

However, on ocean scene types, luminosity thresholding has high precision but low recall, because it often mistakes cloud pixels over ocean for noncloud pixels. Many translucent clouds over ocean or cloud shadows are visually dim, and clouds over the ocean in the tropics may be warm. As such, these warm, dim cloud pixels are frequently misclassified as noncloud. Although luminosity thresholding is a simple and relatively accurate method, its poor performance on ocean and coastline scene types makes it a poor choice for on-orbit cloud detection for BeaverCube-2, given BeaverCube-2's high cloud detection accuracy requirements for oceanic regions.



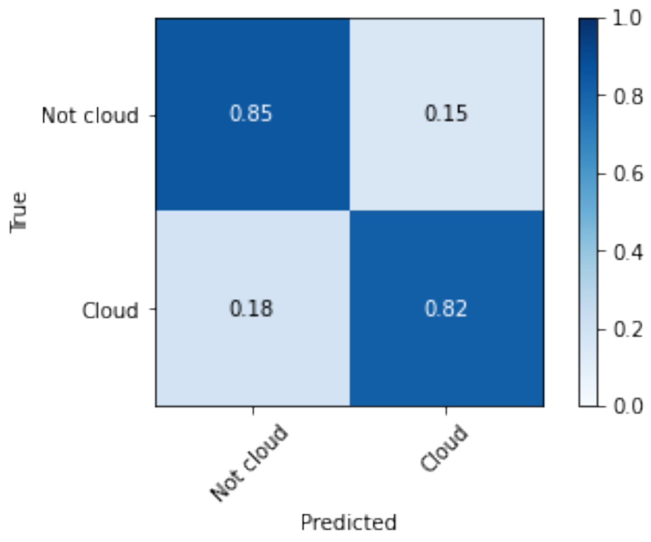


Fig. 10 Receiver operating characteristic curve for luminosity detection

Fig. 11 Confusion matrix for luminosity detection

Table 3 Sensitivity, specificity, accuracy, precision, and recall by image category for luminosity thresholding

Scene Type	Accuracy	Sensitivity	Specificity	Precision	Recall	F_1 Score
Arid	75.76%	94.76%	70.47%	47.17%	94.76%	0.6299
Coastline	77.93%	82.08%	75.28%	68.01%	82.10%	0.7439
Forest	93.63%	81.61%	98.22%	94.60%	81.61%	0.8762
Ocean	85.92%	54.16%	97.54%	88.93%	54.16%	0.6732
Plains	95.17%	93.87%	95.87%	92.42%	93.87%	0.9314
Snow	67.81%	95.91%	61.86%	34.73%	95.91%	0.5100
Urban	92.31%	73.52%	97.44%	88.69%	73.52%	0.8039
Overall	**84.08%**	**82.27%**	**85.24%**	**73.51%**	**82.27%**	**0.7384**

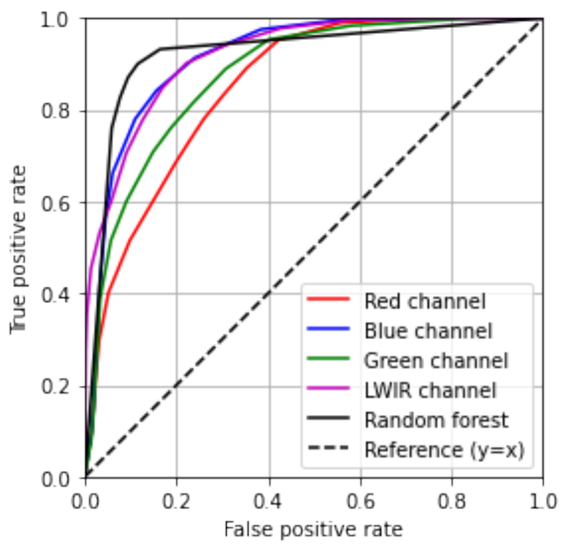
C. Random Forest

We implement our random forest model using the scikit-learn library for machine learning in Python [22]. We use a forest size of 25 trees and a maximum tree depth of 25. We use 80% of our dataset for training and 20% for testing, with the same train/test split as the luminosity-based and U-Net based methods.

Our random forest classifier shows definitive improvements over the luminosity-based classifier, achieving a sensitivity of 87.4%, a specificity of 91.2%, and an overall accuracy increase of 89.6% compared to 84.1%. This is likely because the random forest classifier uses a 3×3 kernel centered on each pixel for classification. This allows 36 features representing the luminosity in the red, green, blue, and IR channels for a pixel and its neighborhood to be used during classification, while our luminosity-based method relies on just 4 features, or the red, green, blue, and IR luminosities of a single pixel.

Each feature in our random forest has an associated Gini importance, or the total reduction in Gini impurity associated with that feature, summed over all nodes and all trees. In a forest with n trees of depth d, with $2^d - 1$ nodes for each tree, the Gini importance for a feature can be expressed as [26]:

$$I_g(f) = \sum_{t=1}^{n} \sum_{j=1}^{2^{d-1}} \Delta i_g(f, j, t) \tag{9}$$



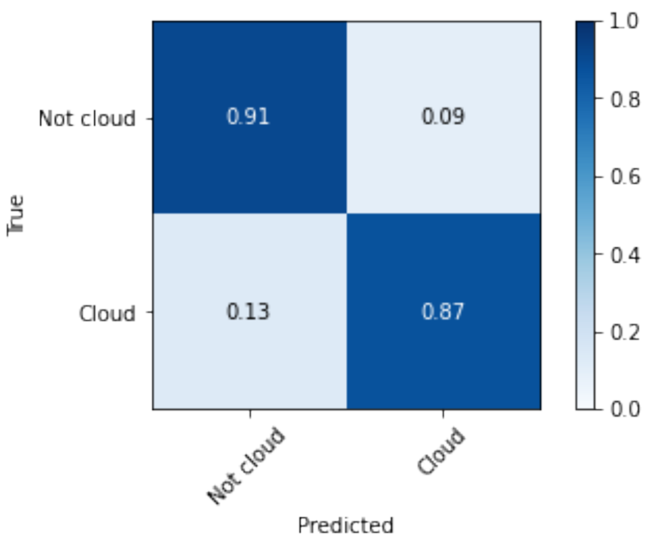
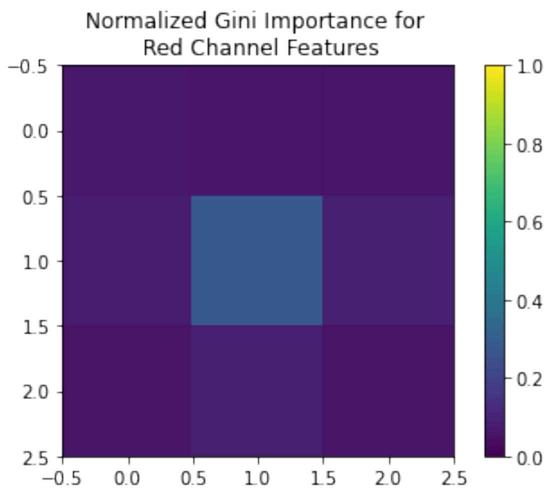


Fig. 12 Receiver operating characteristic curve for random forest detection

Fig. 13 Confusion matrix for random forest detection

We divided the Gini importance of each feature by the Gini importance of the most important feature, and plotted texture maps of the Gini importance for each channel, as shown in Figures 14, 15, 16, and 17.



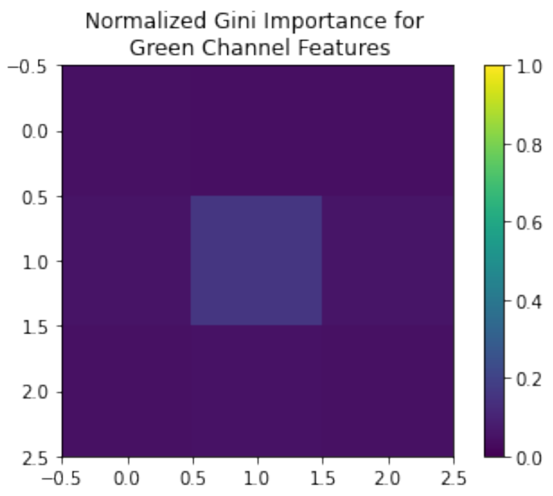


Fig. 14 Feature importance for red channel, with pixel being classified centered at (1,1)

Fig. 15 Feature importance for green channel, with pixel being classified centered at (1,1)

The feature importance heatmaps in Figures 14, 15, 16, and 17 clearly show that most of the Gini impurity decrease in the random forest is associated with features in the blue and LWIR channels, with some smaller amount of Gini impurity decrease associated with the kernel centers of the red and green channels. This result fits with our prior finding that the blue and LWIR channels carry more cloud-related information than the green and red channels, as shown in Figure 10, and demonstrates that our random forest model was able to learn which channels are most useful for cloud identification.

However, the finding that an off-center pixel luminosity in the blue channel, rather than the center of the kernel, is the feature that decreases Gini impurity the most is unexpected. This could be explained by the small size of our random forest (25 trees) – adding more trees to our classifier would increase training and classification times but would reduce variance [22].

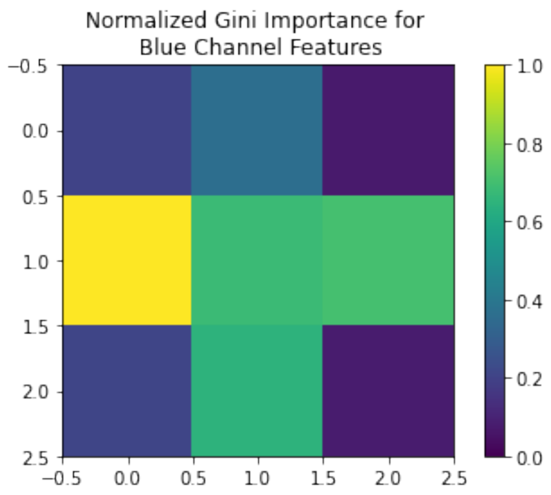


Fig. 16 Feature importance for blue channel, with pixel being classified centered at (1, 1)

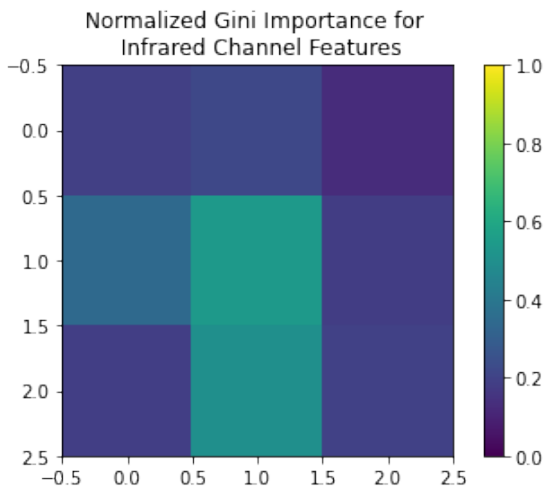


Fig. 17 Feature importance for IR channel, with pixel being classified centered at (1, 1)

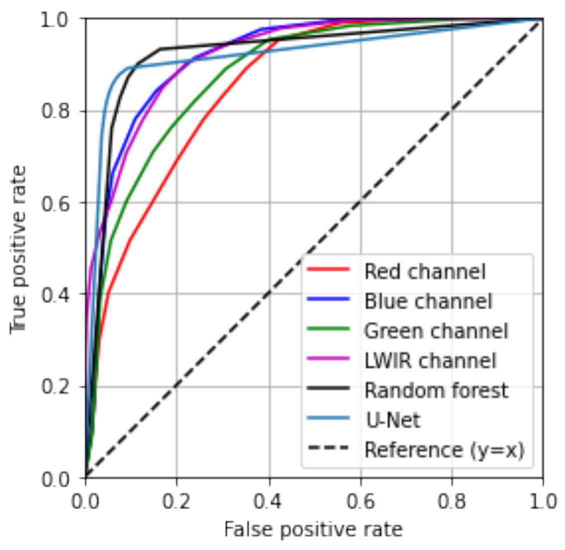
Table 4 Sensitivity, specificity, and accuracy by scene type for random forest detection

Scene Type	Accuracy	Sensitivity	Specificity	Precision	Recall	F_1 Score
Arid	92.01%	99.45%	89.94%	73.33%	99.45%	0.8442
Coastline	93.16%	91.87%	93.99%	90.73%	91.87%	0.9130
Forest	93.95%	80.84%	98.95%	96.71%	80.84%	0.8806
Ocean	87.93%	63.52%	96.85%	88.04%	63.52%	0.7380
Plains	95.72%	91.32%	98.07%	96.21%	91.32%	0.9370
Snow	69.42%	98.25%	63.32%	36.16%	98.25%	0.5286
Urban	95.22%	87.05%	97.45%	90.31%	87.05%	0.8865
Overall	**89.63%**	**87.47%**	**91.22%**	**81.64%**	**87.47%**	**0.8183**

The random forest method shows improved accuracy over the luminosity thresholding method in every category, with notable improvements on arid scene types (92.01% accuracy for random forest and 75.76% accuracy for luminosity thresholding) and coastline scene types (93.16% accuracy for random forest and 77.93% accuracy for luminosity thresholding). However, the random forest method still shows low recall (63.52%) on ocean scenery and low precision (36.16%) on snow scenery, indicating that the random forest method still frequently misclassifies snow pixels as clouds, and misclassifies tropical clouds as ocean.

D. U-Net Based Implementation

The U-Net based model was implemented in PyTorch. The model was first trained for 10,000 epochs to understand how the training loss and validation loss varied with epoch number, and trained for a second time to 700 epochs to the minimize the validation loss and prevent the model from overfitting to data. The deep learning implementation showed modest improvements over the random forest model, with 91.74% accuracy compared to 89.63%, and an improvement in specificity from 91.22% to 94.59%. The largest gains in accuracy were in snow scenes, with accuracy increasing from 69.26% to 85.51%, and precision increasing from 36.16% to 55.46%. This is most likely due to how the cascading convolution and max pooling layers can infer a much larger understanding of context and texture around a pixel, identifying not only clouds, but using shadows and texture to strengthen inference. However, there is still a higher proclivity for false negatives, as evidenced by sensitivity lagging behind specificity.



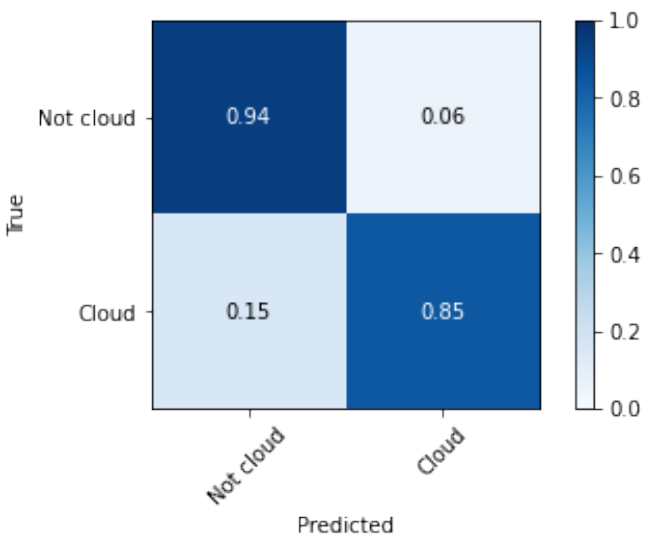


Fig. 18 Receiver operating characteristic curve for deep learning based detection

Fig. 19 Confusion matrix for deep learning based detection

Table 5 Sensitivity, specificity, and accuracy by scene type for U-Net based detection

Scene Type	Accuracy	Sensitivity	Specificity	Precision	Recall	F_1 Score
Arid	93.32%	90.52%	94.10%	81.01%	90.52%	0.8550
Coastline	90.95%	83.03%	96.01%	93.01%	83.03%	0.8773
Forest	94.33%	88.61%	96.51%	90.63%	88.61%	0.8961
Ocean	90.16%	76.74%	95.06%	85.00%	76.74%	0.8066
Plains	93.48%	84.24%	98.43%	96.63%	84.24%	0.9001
Snow	85.51%	85.83%	85.44%	55.46%	85.83%	0.6738
Urban	94.41%	86.40%	96.59%	87.35%	86.40%	0.8687
Overall	**91.74%**	**85.05%**	**94.59%**	**84.16%**	**85.05%**	**0.8397**

E. Comparison

1. Accuracy

In terms of accuracy, luminosity performed the worst, but still at a very reusable average accuracy of 84.08%. Random forest and machine learning methods performed comparatively well with modest improvements going to machine learning, but alternating which scene type each method performed best on.

The two rule-based methods both struggled with snow and ocean scenes as texture and context became much more important, but otherwise remain very usable. This indicates that even a very simple luminosity based scheme is suitable for missions that do not any images of snow or coastline scenes.

The scene type most relevant to that of BeaverCube-2 is that of coastlines and random forest shows a significant performance gain in such scenes, however, the U-Net based implementation has a modestly higher overall accuracy.

Table 6 Comparison of accuracy and sensitivity for all three classifiers across all scene types with a threshold of 0.5.

Scene Type	Accuracy			Sensitivity		
	Luminosity	Random Forest	U-Net	Luminosity	Random Forest	U-Net
Arid	75.76%	92.01%	**93.32%**	94.76%	**99.45%**	90.52%
Coastline	77.93%	**93.16%**	90.95%	82.08%	**91.87%**	83.03%
Forest	93.63%	93.95%	**94.33%**	81.61%	80.84%	**88.61%**
Ocean	85.92%	87.93%	**90.16%**	54.16%	63.52%	**76.74%**
Plains	95.17%	**95.72%**	93.48%	**93.87%**	91.32%	84.24%
Snow	67.81%	69.42%	**85.51%**	95.91%	**98.26%**	85.83%
Urban	92.31%	**95.22%**	94.41%	73.52%	**87.05%**	86.40%
Overall	84.08%	89.63%	**91.74%**	82.27%	**87.47%**	85.05%

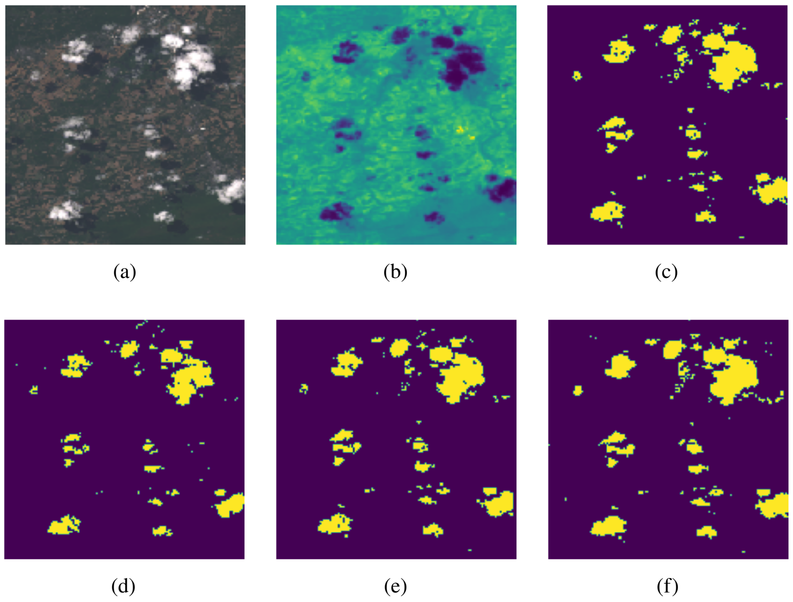


Fig. 20 Comparison of outputs from each model for an easy sample: (a) RGB; (b) LWIR; (c) Reference; (d) Luminosity based; (e) Random forest; (f) U-Net

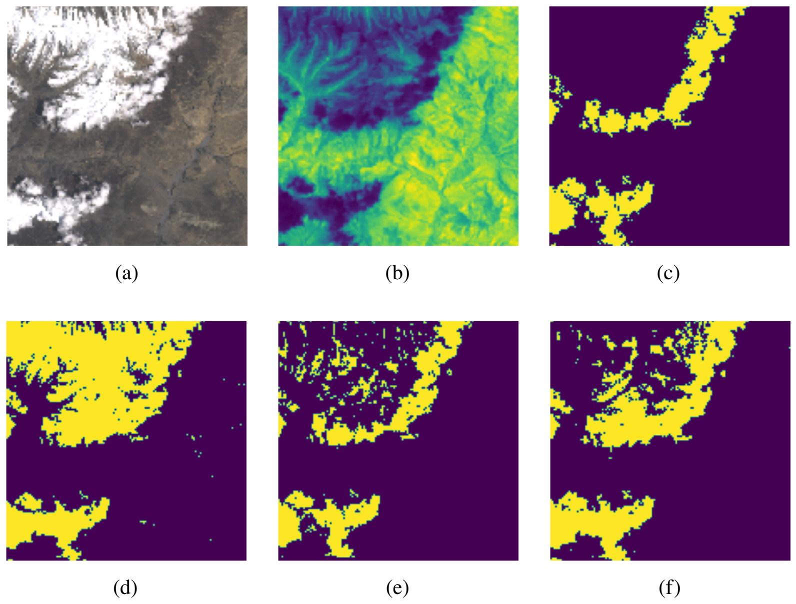


Fig. 21 Comparison of outputs from each model for a hard sample: (a) RGB; (b) LWIR; (c) Reference; (d) Luminosity based; (e) Random forest; (f) U-Net

;

A comparison of all the models, and the input and references images is provided in Figure 20 and Figure 21, split into an easy sample and a hard sample. The easy sample does not have any ambiguous features such as snow, whereas the hard sample has intersecting snow and clouds, which can look nearly identical on RGB and LWIR imagery. All methods perform well in the easy sample, but luminosity-based detection confuses snow for clouds, while the other more sophisticated models are more robust to this class of false-positive, but still confuse a portion of the snow for cloud.

2. Runtime

Table 7 Comparison of runtime performance and peak memory usage for all models to classify a 144×144 pixel image on an Intel Core i7 7700K processor that has 4 cores and a clock speed of 4.2 GHz.

Model	**CPU Runtime [s]**	**Memory Usage [MB]**
Luminosity	0.722	2
Random Forest	1.63	700
U-Net	0.11	1500

At first glance, the performance values for the models in Table 7 seem to show that random forest detection performs significantly slower than our U-Net model. However, the U-Net model was created in PyTorch, which has been extensively optimized for performance. In particular, PyTorch uses lower-level libraries programmed in C++ that make use of SIMD and vectorized instruction sets, such as AVX-512 on modern Intel CPUs [27]. Instead, our random forest and luminosity-based models used only Python code and, as a result, have a significant performance penalty applied to them.

The memory usage of all three methods is significantly different, with luminosity-based detection requiring little to no memory except to store a single image. In contrast, the RF and U-Net models require hundreds of megabytes of memory. U-Net requires an additional 800 MB of memory against a random forest based method for an increase in accuracy of 2%.

VI. Conclusion

At 92% accuracy, the U-Net based method performs comparatively to the random forest based method with an accuracy of 90%, and slightly better than state-of-the-art cloud detection algorithms like Fmask that depend on augmenting visible-band images with data from near-infrared (NIR), short-wave infrared (SWIR), and thermal infrared (TIRS) bands [2, 11]. Our U-Net based method, implemented using the highly optimized PyTorch library, takes only 110 ms

Although neither our U-Net or random forest based methods has not yet been tested on the AI SoC itself, their high accuracy and low classification time show significant promise for efficient and accurate on-orbit cloud identification on BeaverCube-2, and the hardware specifications of the SoC suggest it should perform similarly if not better than the algorithm does on the i7.

In addition, depending on mission requirements and ground coverage areas, our luminosity-based detection method has extremely light computational requirements and performs very well with an overall accuracy of 84%.

VII. Future Work

A. Model Improvements

Our U-Net based model is translationally equivariant, so it is agnostic to the position of clouds in images [25]. However, our U-Net based model is not rotationally equivariant, so its output depends on how clouds in input imagery are rotated with respect to the nadir vector of the spacecraft's cameras. In the future, we plan to develop a roto-translationally equivariant deep learning model based on $SE(2)$ group convolutional neural networks (G-CNNs) for cloud identification [28, 29]. Roto-translationally equivariant G-CNNs are currently used for image segmentation in medical imagery, where features of interest may be rotated or translated with respect to the camera, which is unusual in terrestrial imaging [29]. Using a G-CNN-based architecture will reduce our deep learning model's parameter space and take advantage of orbital geometry and symmetry, allowing our model to identify clouds regardless of their orientation or position with respect to BeaverCube-2's camera. The amount of improvement based on our current method which augments data using random flips is unclear, and will need to be experimentally determined.

Additionally, the rest of the computer vision pipeline must be developed around this cloud segmentation work and tested end to end. This work abstracts away the additional challenges of utilizing multiband on-orbit remote sensing data, such as issues with calibration and image registration.

B. Hardware Integration

The following steps include porting the model to work on the AI SoC and verifying that the implementation works as expected. Additional calibration and radiometric analyses will need to be performed to transfer the weights from the model to run on-orbit, as Landsat 8's optical frontend is very different from BeaverCube-2. Final performance estimation can then be done, along with thermal analysis and verification that the model is not too computationally intensive.

Additionally, the optical frontend of BeaverCube-2 will need to be ported to our dataset, by measuring the point-spread function (PSF) and simulating imagery using Landsat 8 data as ground truth.

Acknowledgment

The work was sponsored by the Northrop Grumman Corporation as part of BeaverCube-2 hardware development.

References

[1] Pereira, P. d. V., Garcia, M., Schroeder, M., Caldelas, H., Lindsay, C., Choi, A., Pfrang, K., Gagnon, A., Meredith, A., McKeen, P., Clark, J., Murphy, T. J., Baber, S., Khan, S., Miller, A., Velez, G., Campbell, M., Coray, J., Koldada, J., Tran, T., Austin, S., Louthain, A., Wells, T., Kabir, M., Sit, E., Haughwout, C., and Cahoy, K., "BeaverCube: Coastal Imaging with VIS/LWIR CubeSats," 2020. URL https://digitalcommons.usu.edu/smallsat/2020/all2020/126.

[2] Zhu, Z., and Woodcock, C. E., "Object-based cloud and cloud shadow detection in Landsat imagery," *Remote sensing of environment*, Vol. 118, 2012, pp. 83–94.

[3] Zupanc, A., "Improving Cloud Detection with Machine Learning," , 2017.

[4] Wagstaff, K. L., Altinok, A., Chien, S. A., Rebbapragada, U., Schaffer, S. R., Thompson, D. R., and Tran, D. Q., "Cloud Filtering and Novelty Detection using Onboard Machine Learning for the EO-1 Spacecraft," *International Joint Conference on Artificial Intelligence*, 2017, p. 4.

[5] Ronneberger, O., Fischer, P., and Brox, T., "U-Net: Convolutional Networks for Biomedical Image Segmentation," , 2015.

[6] Dial, G., Bowen, H., Gerlach, F., Grodecki, J., and Oleszczuk, R., "IKONOS satellite, imagery, and products," *Remote Sensing of Environment*, Vol. 88, No. 1, 2003, pp. 23–36. https://doi.org/https://doi.org/10.1016/j.rse.2003.08.014, URL https://www.sciencedirect.com/science/article/pii/S0034425703002293, iKONOS Fine Spatial Resolution Land Observation.

[7] NASA, "MODIS Web," 1999. URL https://modis.gsfc.nasa.gov/about/.

[8] Stephens, G. L., and Kummerow, C. D., "The Remote Sensing of Clouds and Precipitation from Space: A Review," *Journal of the Atmospheric Sciences*, Vol. 64, No. 11, 2007, pp. 3742–3765. https://doi.org/10.1175/2006JAS2375.1, URL https://journals.ametsoc.org/view/journals/atsc/64/11/2006jas2375.1.xml, publisher: American Meteorological Society Section: Journal of the Atmospheric Sciences.

[9] Knight, E. J., and Kvaran, G., "Landsat-8 Operational Land Imager Design, Characterization and Performance," *Remote Sensing*, Vol. 6, No. 11, 2014, pp. 10286–10305. https://doi.org/10.3390/rs61110286, URL https://www.mdpi.com/2072-4292/6/11/10286, number: 11 Publisher: Multidisciplinary Digital Publishing Institute.

[10] Zhu, Z., Wang, S., and Woodcock, C. E., "Improvement and expansion of the Fmask algorithm: Cloud, cloud shadow, and snow detection for Landsats 4–7, 8, and Sentinel 2 images," *Remote Sensing of Environment*, Vol. 159, 2015, pp. 269–277.

[11] Li, Z., Shen, H., Cheng, Q., Liu, Y., You, S., and He, Z., "Deep learning based cloud detection for medium and high resolution remote sensing images of different sensors," *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 150, 2019, pp. 197–212. https://doi.org/10.1016/j.isprsjprs.2019.02.017, URL https://linkinghub.elsevier.com/retrieve/pii/S0924271619300565.

[12] McClain, C. R., Meister, G., and IOCCG, "Mission Requirements for Future Ocean-Colour Sensors." Report, International Ocean Colour Coordinating Group (IOCCG), 2012. https://doi.org/10.25607/OBP-104, URL https://repository.oceanbestpractices.org/handle/11329/524, accepted: 2018-09-26T12:37:32Z.

[13] "mvBlueFOX-IGC Series," , ???? URL https://publications.balluff.com/pdfengine/pdf?id=MP11708989&language=en&type=kmat.

[14] "Edumund Optics - 442 nm CWL 25mm Diameter Filter, OD = 4.0, FWHM = 10 ± 2 nm," , ???? URL https://www.edmundoptics.com/p/442nm-cwl-25mm-dia-hard-coated-od-4-10nm-bandpass-filter/19790/.

[15] "1/1.8 16mm F/1.4 with Lock for Mega Pixel Cameras," , ???? URL https://www.tamron-usa.com/IO/IndOp/prod/m118fm16.php.

[16] FLIR, "FLIR Boson - Compact LWIR Thermal Camera," , 2021. URL https://flir.netx.net/file/asset/12673/original/attachment.

[17] FLIR, "FLIR Boson Thermal Imaging Core - Datasheet," , 2021. URL https://flir.netx.net/file/asset/15754/original/attachment.

[18] Anan Ju, L. D. F. Z. X. Z. X. P. H. Z. J. B., Hongxia Guo, "Analysis of Ion-Induced SEFI and SEL Phenomena in 90nm NOR Flash Memory," *IEEE Transactions on Nuclear Science*, 2021, pp. 2508–2516.

[19] Mohajerani, S., and Saeedi, P., "Cloud-Net+: A Cloud Segmentation CNN for Landsat 8 Remote Sensing Imagery Optimized with Filtered Jaccard Loss Function," *arXiv:2001.08768 [cs, eess]*, 2020. URL http://arxiv.org/abs/2001.08768, arXiv: 2001.08768.

[20] "What are the band designations for the Landsat satellites?" Fact Sheet, 2018. URL https://www.usgs.gov/faqs/what-are-band-designations-landsat-satellites, series: Fact Sheet.

[21] Rodriguez-Galiano, V. F., Ghimire, B., Rogan, J., Chica-Olmo, M., and Rigol-Sanchez, J. P., "An assessment of the effectiveness of a random forest classifier for land-cover classification," *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 67, 2012, pp. 93–104.

[22] Louppe, G., "Understanding random forests: From theory to practice," *arXiv preprint arXiv:1407.7502*, 2014.

[23] Wager, S., Hastie, T., and Efron, B., "Confidence intervals for random forests: The jackknife and the infinitesimal jackknife," *The Journal of Machine Learning Research*, Vol. 15, No. 1, 2014, pp. 1625–1651.

[24] Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T., "Visualizing the Loss Landscape of Neural Nets," , 2018.

[25] Zhang, R., “Making Convolutional Networks Shift-Invariant Again,” , 2019.

[26] Menze, B. H., Kelm, B. M., Masuch, R., Himmelreich, U., Bachert, P., Petrich, W., and Hamprecht, F. A., “A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data,” *BMC bioinformatics*, Vol. 10, No. 1, 2009, pp. 1–16.

[27] “Intel and Facebook* collaborate to boost PyTorch* CPU performance,” , 2019. URL https://www.intel.com/content/www/us/en/develop/articles/intel-and-facebook-collaborate-to-boost-pytorch-cpu-performance.html.

[28] Cohen, T., and Welling, M., “Group Equivariant Convolutional Networks,” *Proceedings of The 33rd International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 48, edited by M. F. Balcan and K. Q. Weinberger, PMLR, New York, New York, USA, 2016, pp. 2990–2999. URL https://proceedings.mlr.press/v48/cohenc16.html.

[29] Bekkers, E. J., Lafarge, M. W., Veta, M., Eppenhof, K. A., Pluim, J. P., and Duits, R., “Roto-translation covariant convolutional networks for medical image analysis,” *International conference on medical image computing and computer-assisted intervention*, Springer, 2018, pp. 440–448.