Machine Learning Image Processing Algorithms Onboard OPS-SAT

Shreeyam Kacker[1], Alex Meredith[1], Georges Labrèche[2], Kerri Cahoy[1]

1 - Massachusetts Institute of Technology
2 - Tanagra Space

36th annual AIAA/USU Conference on Small Satellites

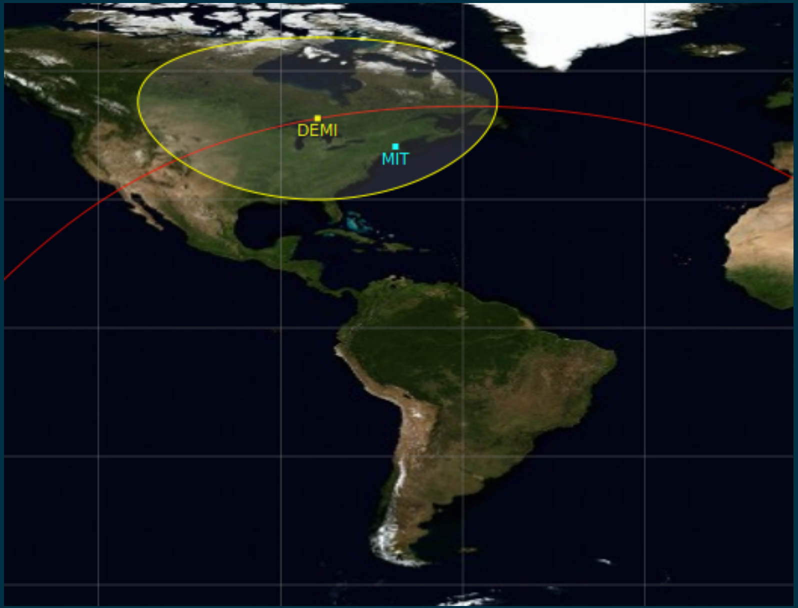
Agenda

- Introduction
  - Motivation
  - BeaverCube-2 Introduction
  - OPS-SAT Introduction
- Approach and Algorithms
- On-orbit Results
- Conclusions & Future Work

Introduction

Motivation

- Goal: Move image processing on-orbit instead of ground
  - Downlink bandwidth **constrained**
  - Prioritize more **useful data** to ground
- CubeSat resources are constrained by **compute power** and **heat dissipation**
- **De-risking** algorithms by deploying on OPS-SAT first



Example overpass over MIT ground station

BeaverCube-2 Introduction (1/3)

- BeaverCube 2 is a 3U CubeSat that aims to image coastal areas of Cape Hatteras and **identify ocean fronts**
- Cloud segmentation an intrinsic part of remote sensing image processing
- BC-2 has AI accelerator system-on-a-chip (SoC) and two visible-spectrum cameras and one long-wave infrared (LWIR) camera
- **Cloud-free or low-cloud** (<5% cloud) images for coastline and front identification and downlink

Cape Hatteras, North Carolina

BeaverCube-2 CONOPS (2/3)

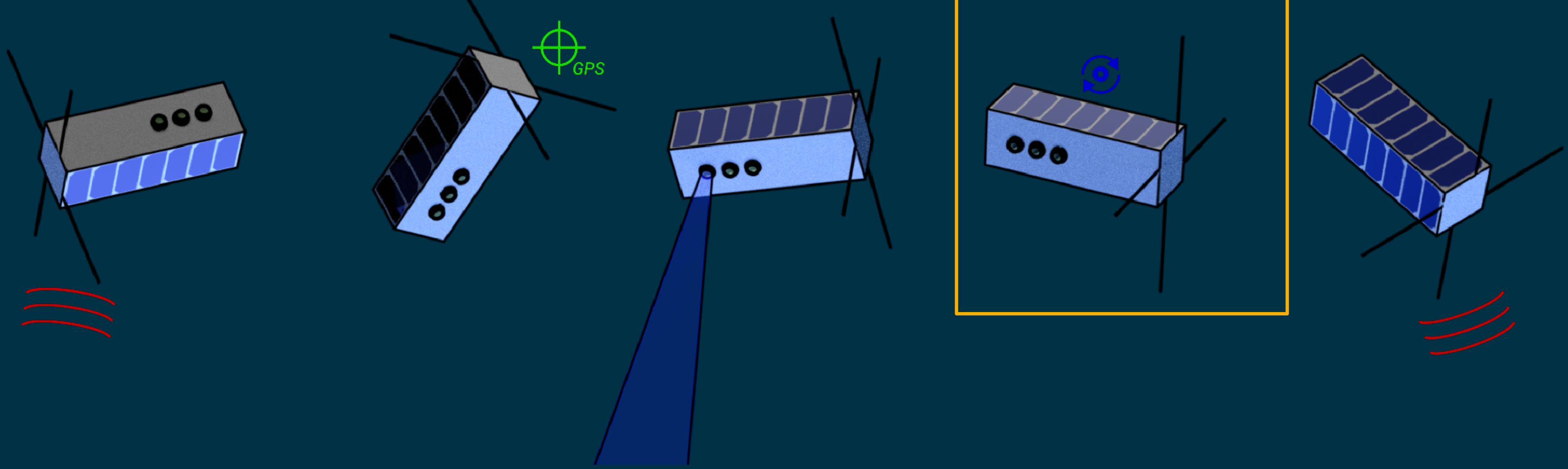
1. Receive Command
Ground command specifies location to capture image

2. Target Pointing
Adjust spacecraft attitude to point camera at desired target

3. Capture Image
Cameras capture and store images of target location

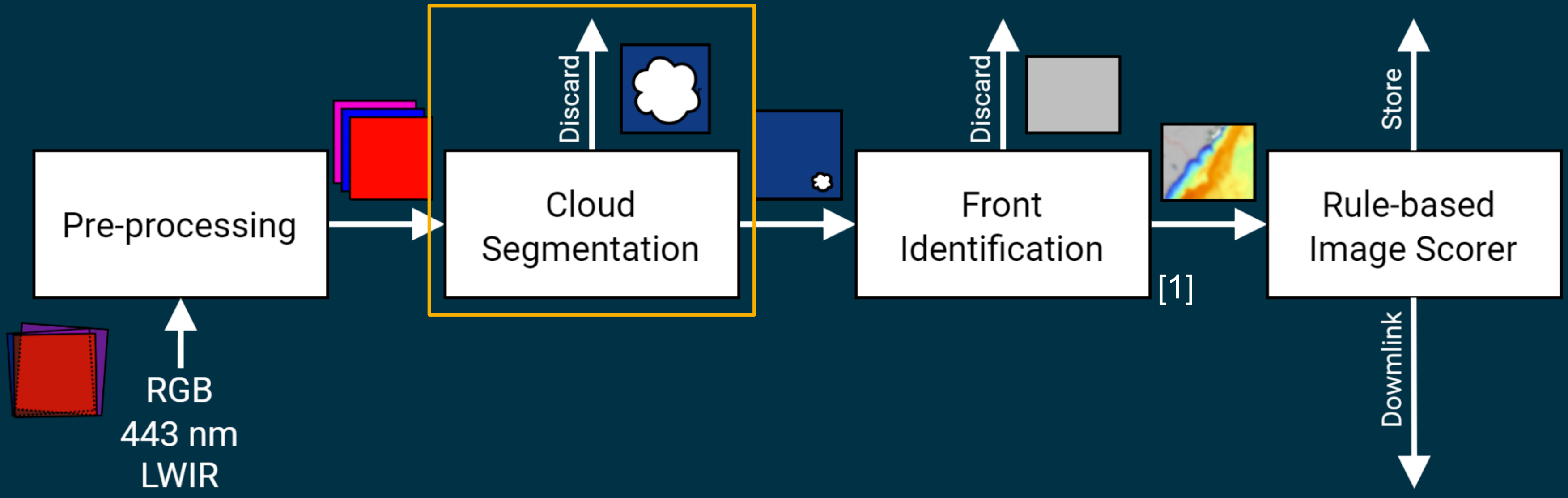
4. Process Image
Onboard processing of image to identify ocean fronts

5. Downlink
Downlink data on overpasses of MIT ground station



(Shreeyam Kacker, MIT)

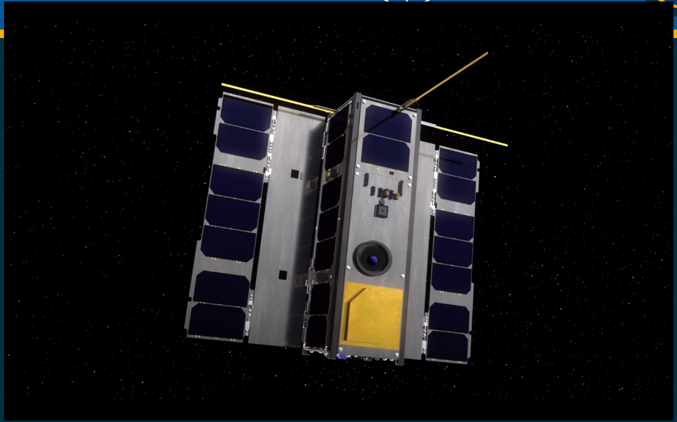
BeaverCube-2 Image Pipeline (3/3)



[1] V. C. Felt, *Machine Learning Models for On-Orbit Detection of Temperature and Chlorophyll Ocean Fronts (2022),* SM thesis. Available starlab.mit.edu

OPS-SAT

- 3U CubeSat launched on December 2019
- Ground station is SMILE at ESOC in Darmstadt
- **Open experimenter platform** allows researchers from all over the world run their experiment onboard OPS-SAT
- RGB camera + Intel Cyclone V FPGA with two hard ARM A9 cores at 800 MHz on the Satellite Experimental Processing Platform (SEPP)



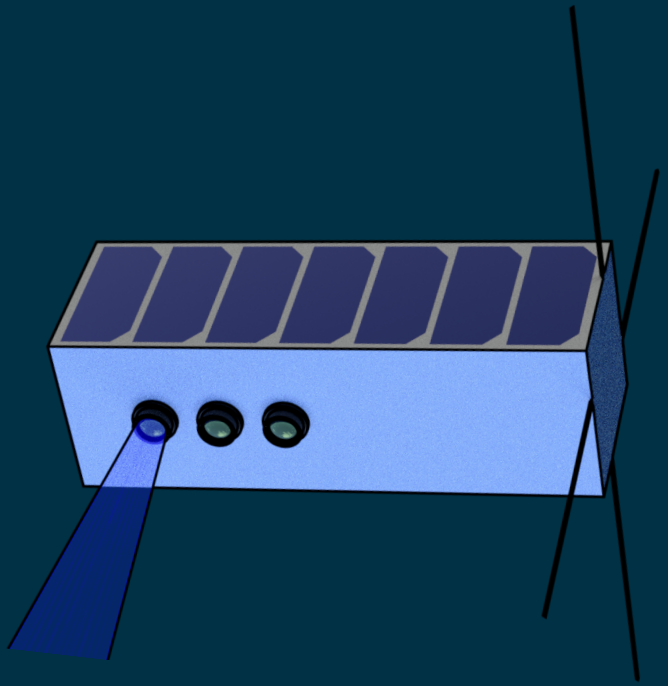
Artist's impression of OPS-SAT

SMILE ground station

Approach and Algorithms

Space Constraints

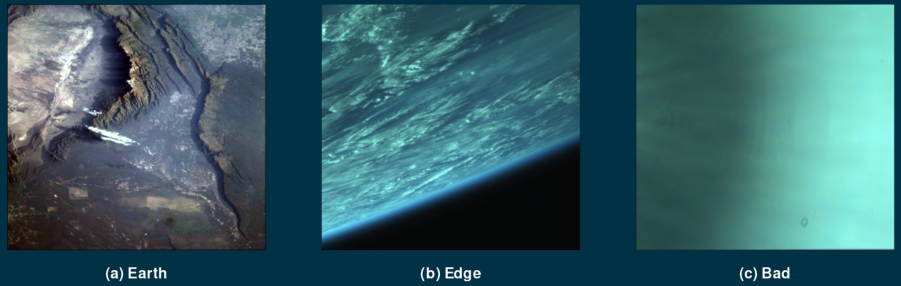
- Resource constrained
  - Computation limited by heat and power draw
  - Constraints on disk footprint set by uplink ground station
- Orientation
  - No specific nadir rotation
  - Unlike self-driving cars
- Radiation
  - Bit flips



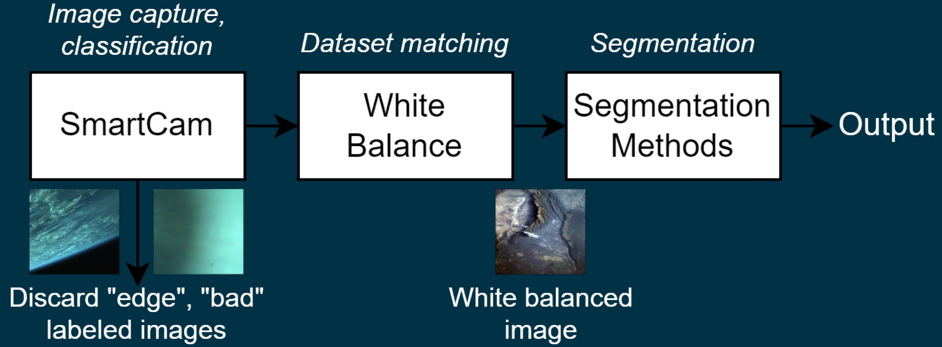
Rendering of BeaverCube-2

OPS-SAT Image Pipeline

- Two stage pipeline: Classification and segmentation
- SmartCam resizes the input and uses a lightweight classifier to detect Earth images
- White balancing matches input to Landsat 8 training data



SmartCam output labels



OPS-SAT computer vision pipeline

Approach - Luminosity Thresholding 💡

- Luminosity-based thresholding classifies pixels based on their luminosity in the red, green and blue channels
- Blue channel is the **most informative** for cloud identification
- **Only one vote** is needed to classify a pixel as a cloud
- Implemented pixel-wise in C++

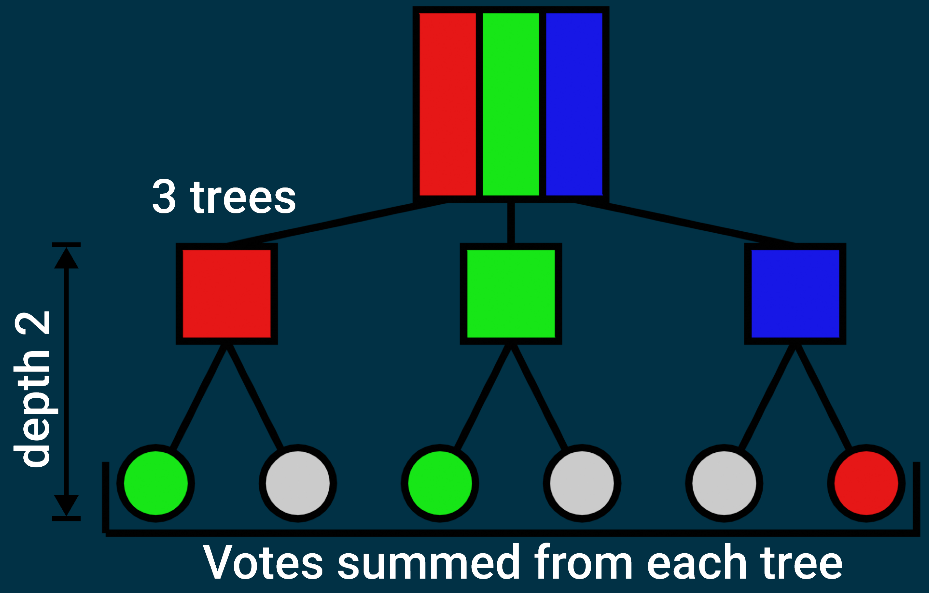


Diagram of luminosity thresholding method

Approach - Random Forest 🎄

- **Kernel-based** random forest produces a binary tree representation
- **Multiple trees** used to form a forest
- Trees make decisions based on features of the data
- We used 10 trees, with a max tree depth of 10
- Implemented using Ranger library with custom uint8 mode implemented

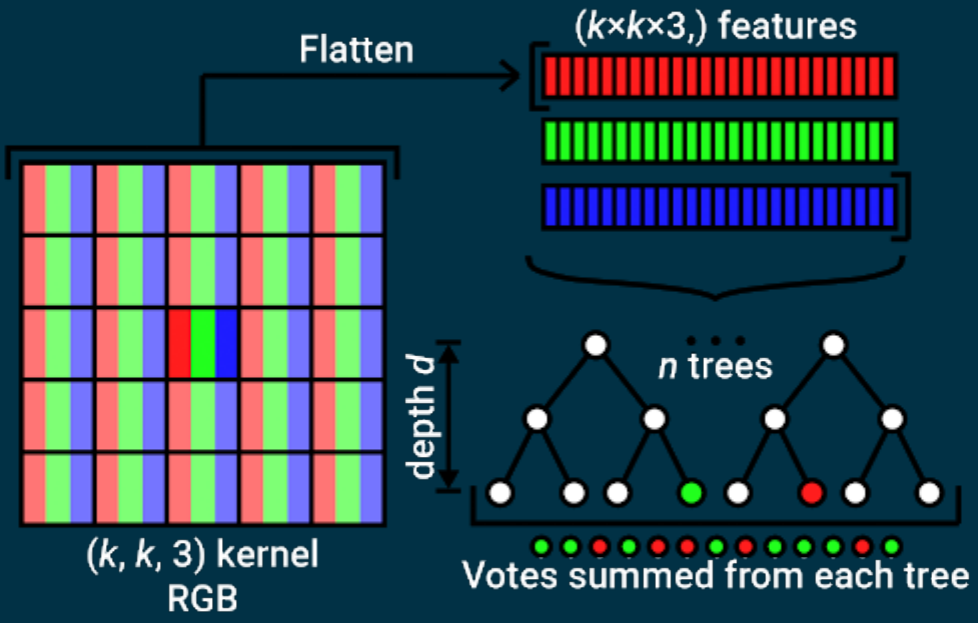


Diagram of random forest method

Approach - Deep Learning 🧠

- Selected **U-Net based architecture** for image segmentation
- Skip connections allow for faster training and **more precise** segmentation boundaries
- **Focal loss** used to improve performance in 100% cloudy/not cloud scenes
- Implemented using TensorFlow Lite, easily ported to BC-2's AI SoC

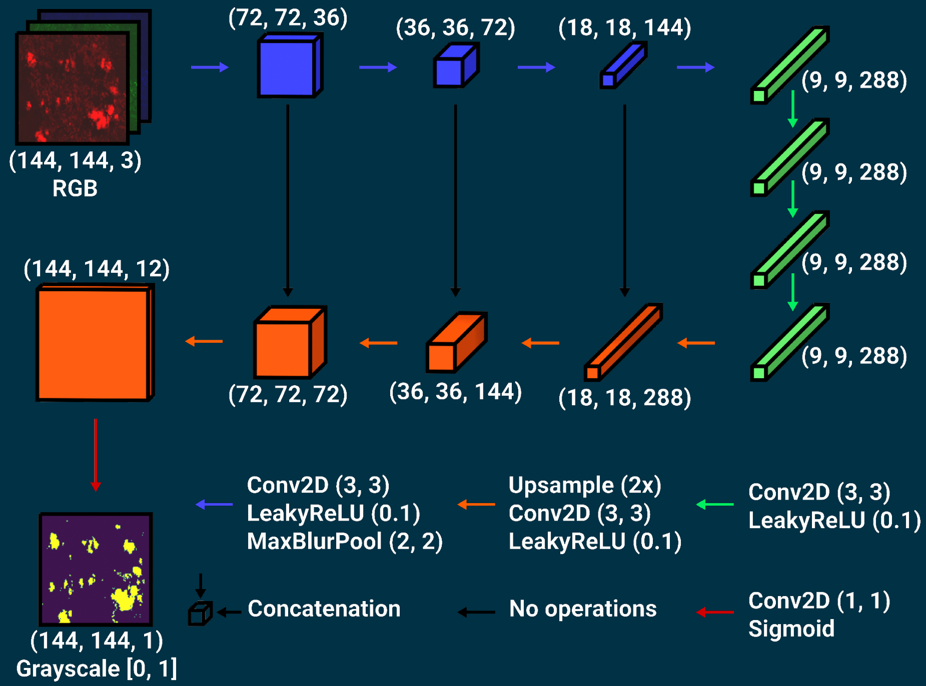
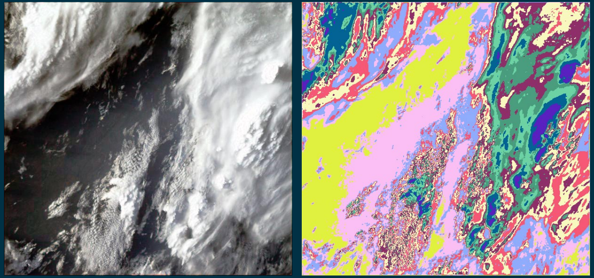


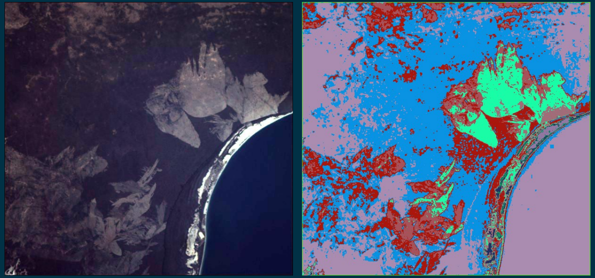
Diagram of U-Net architecture

Approach - K-Means Clustering ✨

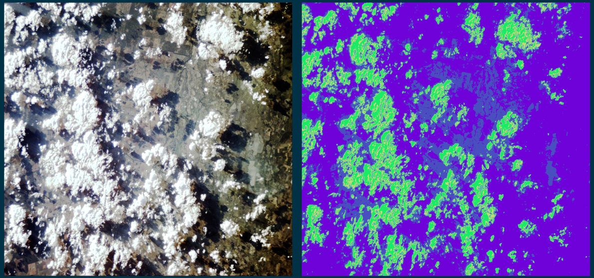
- Minimum-entropy clustering using kmeans++ algorithm
- K-value variable from 2 to 11
    - k=2 used for cloud segmentation, k=11 limited determined from memory constraints
- Implemented in C++ using dkm k-means clustering library
- Implemented by OPS-SAT flight control team (FCT) for comparison



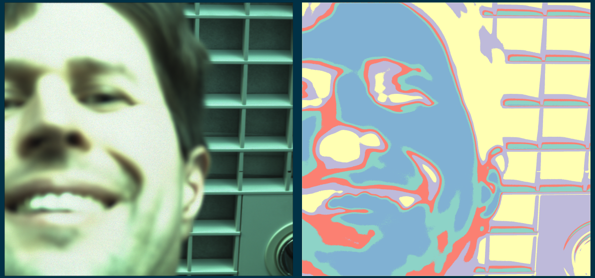
Cumulus clouds (k=10)



Clear sky with coastline (k=10)



Stratos clouds (k=5)



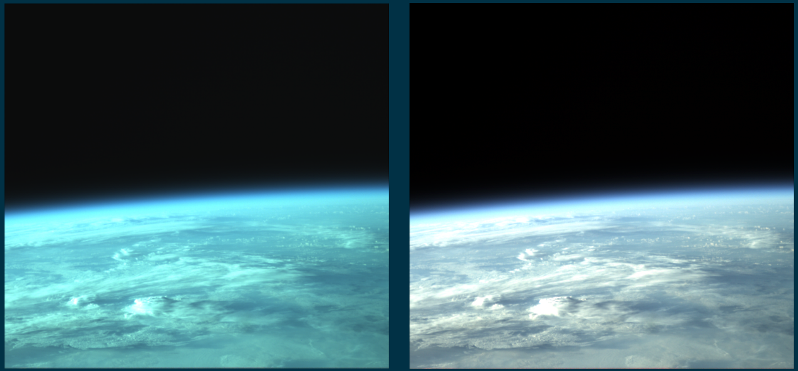
Vladimir from OPS-SAT FCT (k=5)

Orbital results of k-means clustering on varying samples with varying k value

On-orbit Results

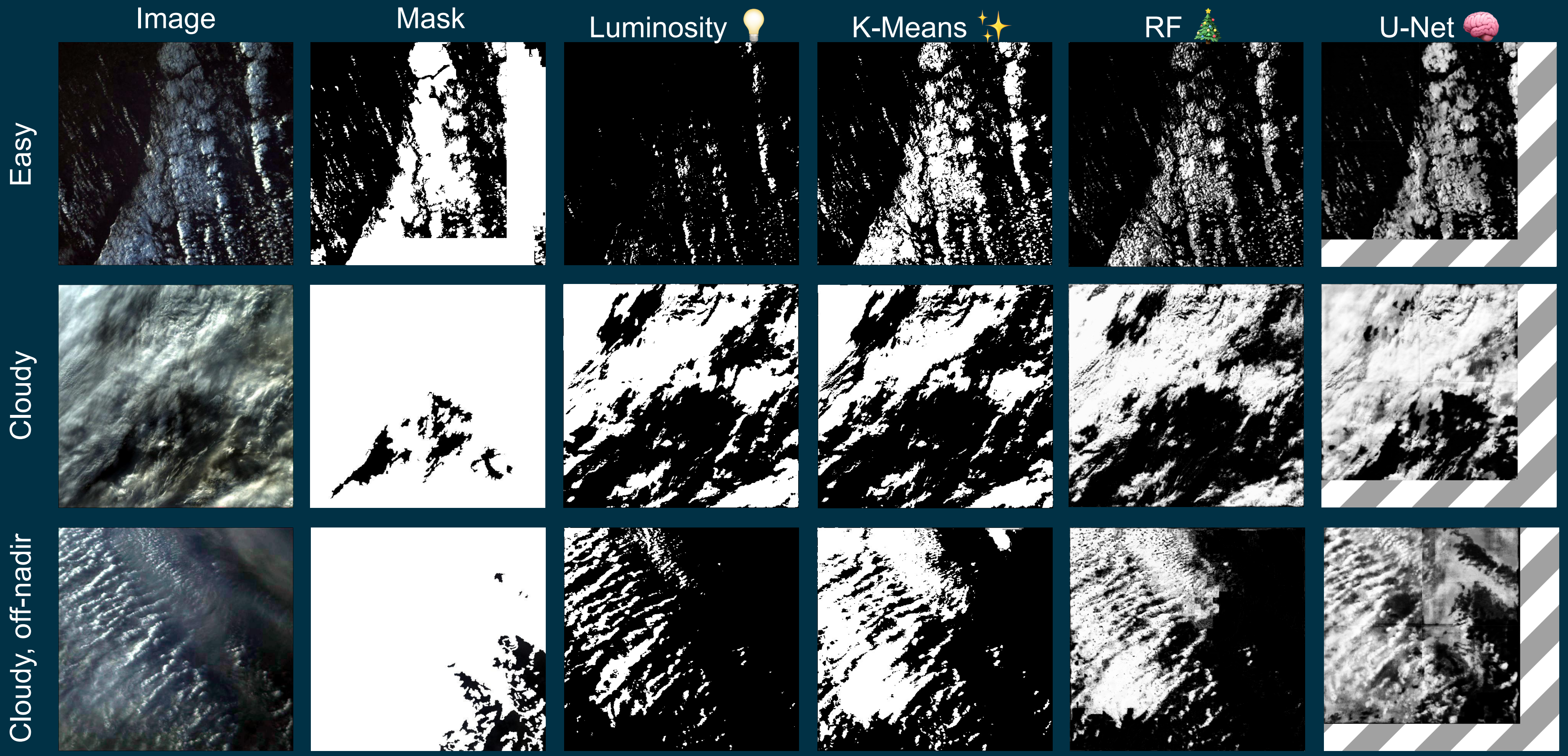
Lessons Learned

- Originally used 25 trees with a max depth of 25
  - Resulted in a 287 MB forest! Cut down to 10 trees with a max depth of 10 in custom uint8 mode resulting in a 50 MB compressible forest
- White balancing initially not implemented
  - Experiment showed white balancing was essential
  - Poor performance without white balancing
- Random forest testing conducted on EM
  - Radiation fault on OPS-SAT



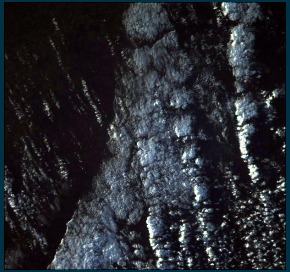
Comparison of raw (left) and white-balanced (right) images

Model Performance (Comparative)

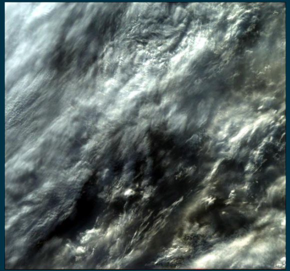


Model Performance (Quantitative)

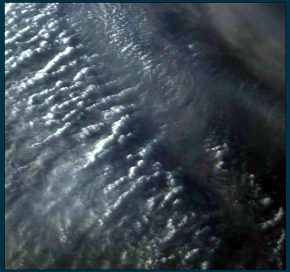
Easy



Cloudy



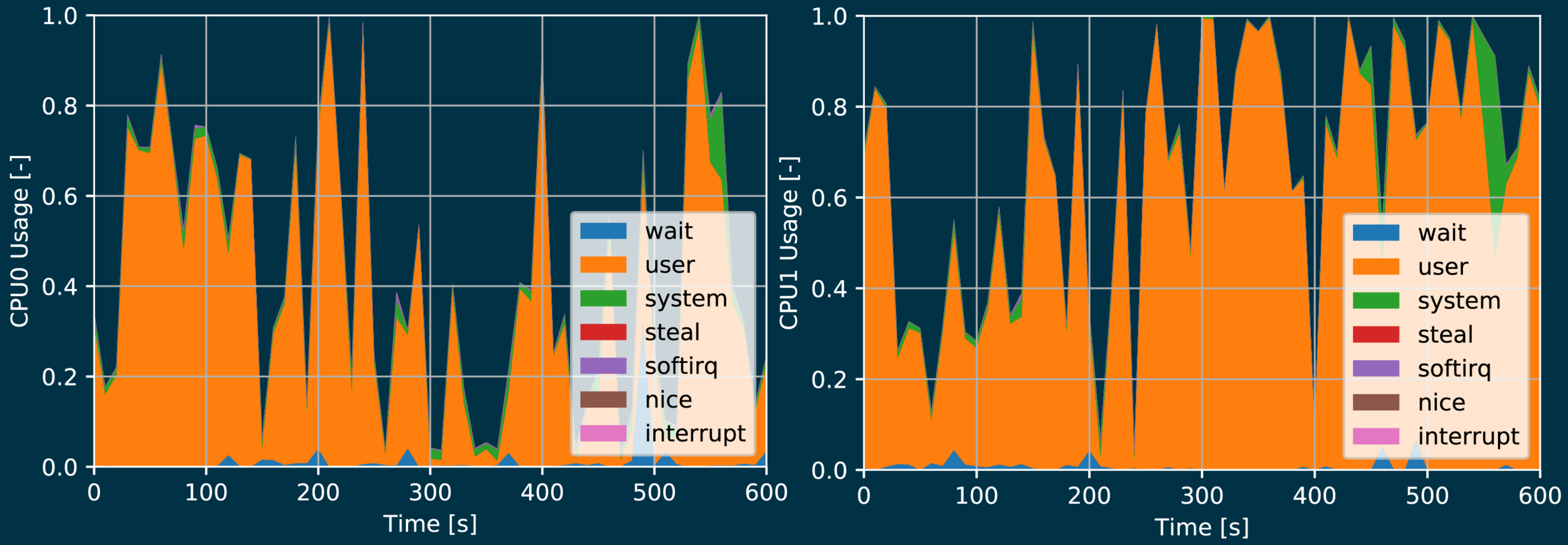
Cloudy, off-nadir



	Accuracy	Balanced Accuracy	Sensitivity	Specificity	Precision	Recall	F_1 Score
Luminosity 💡	0.22	0.56	0.12	1.00	1.00	0.12	0.21
K-Means ✨	0.40	0.66	0.32	1.00	1.00	0.32	0.49
Random Forest 🎄	0.39	0.66	0.31	1.00	1.00	0.31	0.48
U-Net 🧠	0.62	0.78	0.57	1.00	1.00	0.57	0.73

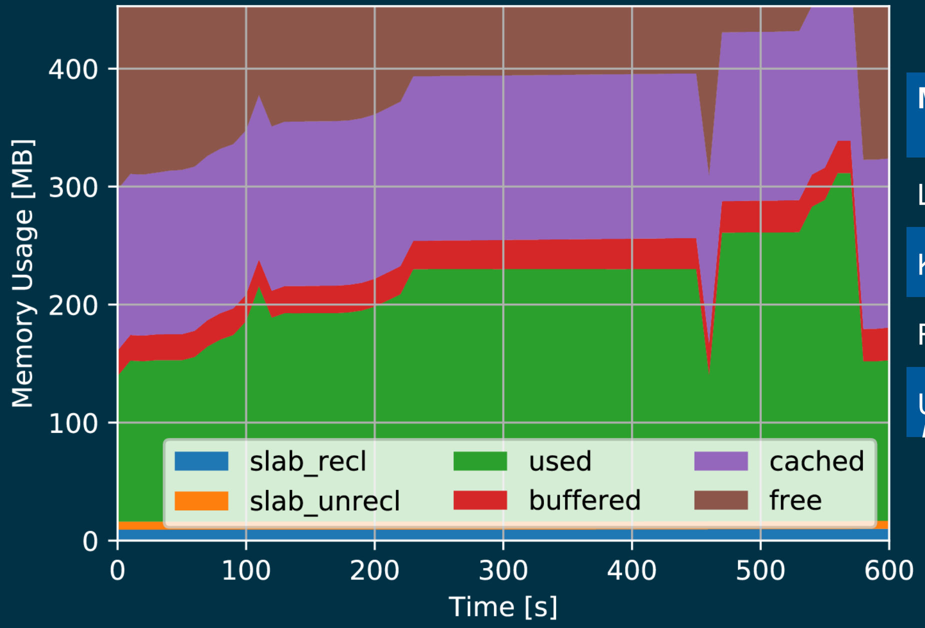
	Accuracy	Balanced Accuracy	Sensitivity	Specificity	Precision	Recall	F_1 Score
Luminosity 💡	0.60	0.78	0.57	1.00	1.00	0.57	0.73
K-Means ✨	0.51	0.74	0.47	1.00	1.00	0.47	0.64
Random Forest 🎄	0.64	0.80	0.60	1.00	1.00	0.60	0.75
U-Net 🧠	0.79	0.89	0.77	1.00	1.00	0.77	0.87

Resource Utilization (CPU, Memory, Disk)



CPU usage on core 0 (left) and core 1 (right) during experiment operation

Resource Utilization (Memory, Disk)

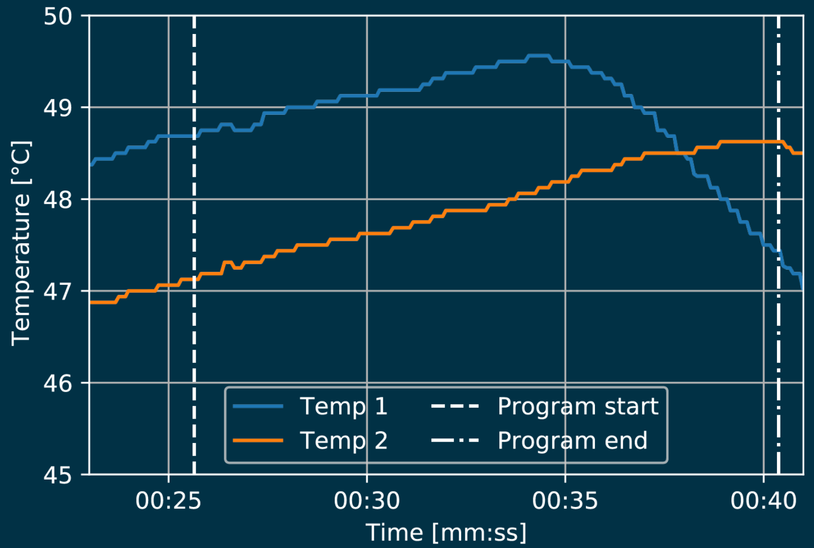


Memory usage during experiment operation

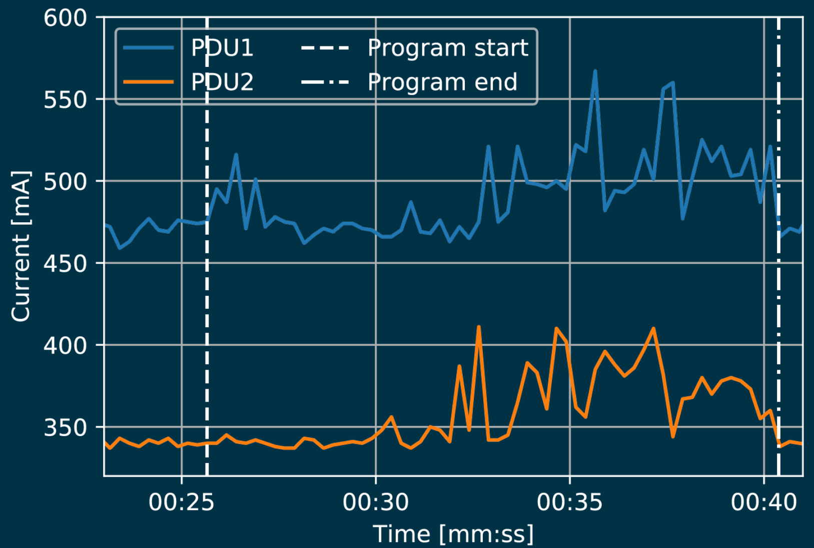
Model	Peak Memory [MB]	Disk Footprint [MB]	Runtime [mm:ss]
Luminosity 💡	N/A	N/A	00:02
K-Means ✨	5	N/A	00:03
Random Forest 🎄	170	48.3	01:44
U-Net 🧠	90	3.3	07:13

Peak memory usage, model disk footprint, and runtime for each algorithm during experiment operation

Temperature and Power Usage



Temperature on SEPP during single experiment run



Current draw from SEPP during single experiment run

Model	Energy [mWh]
Luminosity 💡	0.3
K-Means ✨	0.5
Random Forest🎄	12.6
U-Net 🧠	33.6

Energy required per inference for different models

Conclusions & Future Work

- Accuracy on cloudy images is most important despite diminishing returns on compute power
  - Improved performance of more complex methods is desirable
  - Motivates the use of the AI SoC on BeaverCube-2
  - Focal loss helps a lot
- Additional steps required to have useful transfer of training data
  - White balancing
  - Off-nadir scenes
- Thermals are well controlled on OPS-SAT
- Data driven methods can grow out of control in disk space fast
- Developing scheduling algorithms to respond to cloudy images and revisit them autonomously

Thank you!

Contact: shreeyam@mit.edu / ameredit@mit.edu