



MISSION CONTROL

**A Comparative Study of High-Level and Low-Level
Implementations of Deep Learning Models for Spacecraft
Final Review Meeting**

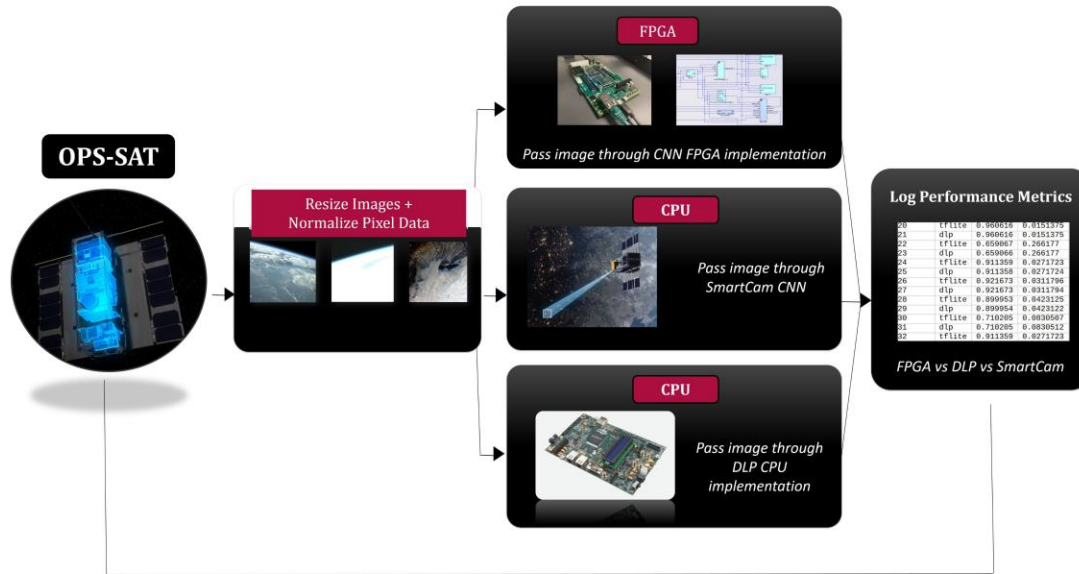
ESA OPS-SAT OSIP Campaign

Idea: I-2021-03755

December 16th, 2022

MCSS.2114

OPS-SAT



“We will perform the first comparative study to make use of NNEF for deep learning on a spacecraft by comparing our low-level implementation of the OPS-SAT SmartCam model against the existing high-level model that uses the Tensorflow Lite C API. The software developed as part of our study will improve the operational performance of SmartCam and provide a modular scaffold for future space-based deep learning FPGA technology.”

Team



Yolanda Brown, PMP
Project Manager



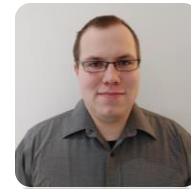
Luis Chavier
Senior Software
Developer



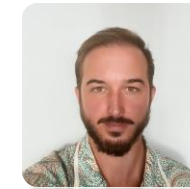
Hugo Burd
Software Engineer



Becca Bonham-Carter
Embedded Software
Engineer



Tim Heydrich
Junior Software
Engineer



Galen O'Shea
Junior AI Specialist



Dr. Andrew Macdonald
AI Product Owner



Nevedhaa Ayyappan
Software Engineering Student Intern

Agenda

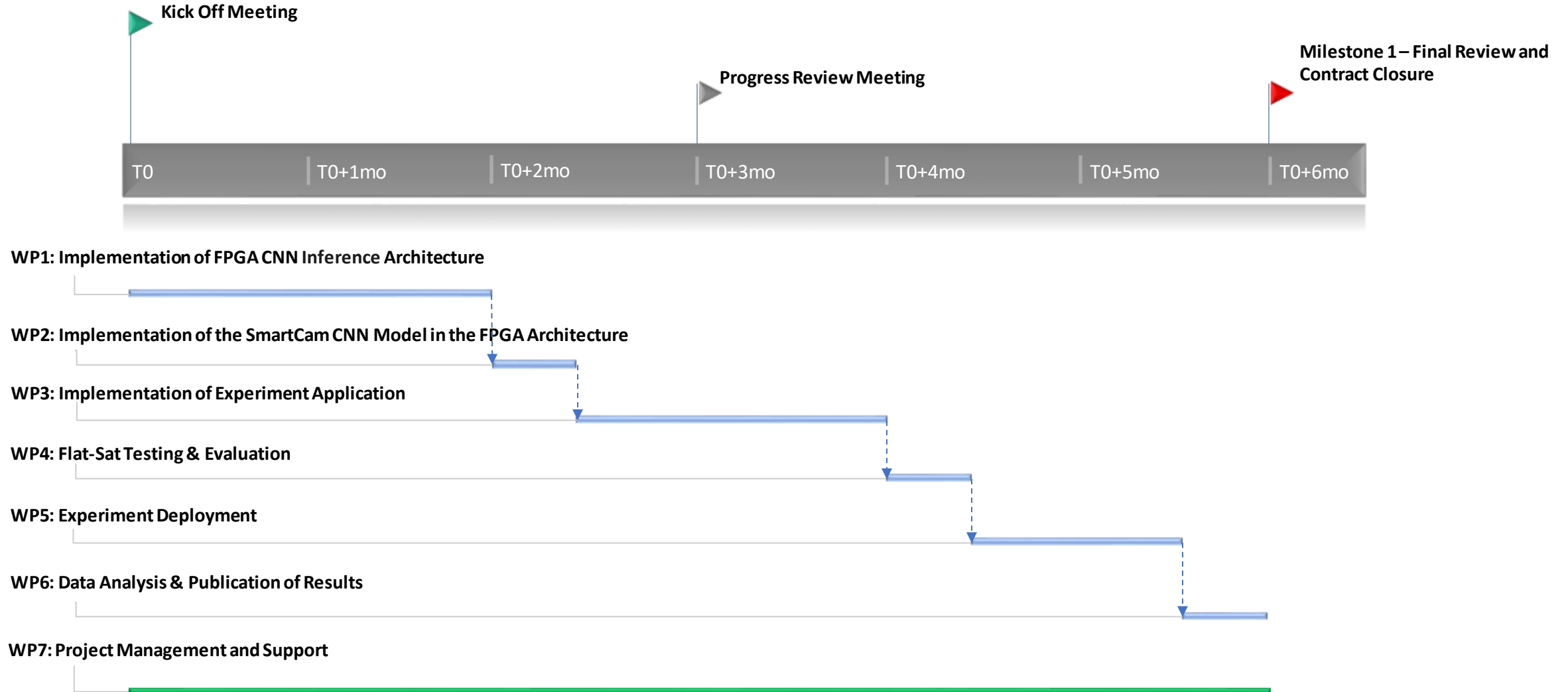
- Programmatic Update
- Technical Update – Experiment 177
 - Intro to Mission Control's Deep Learning Accelerator
 - Hardware Set Up & Initial FPGA Development
 - SmartCam Test Data and Orbital Path Planning
 - Implementation of Experiment
 - Testing, Evaluation, and Deployment
- Conclusions and Next Steps





Programmatic Update

Schedule



Project and Risk Management

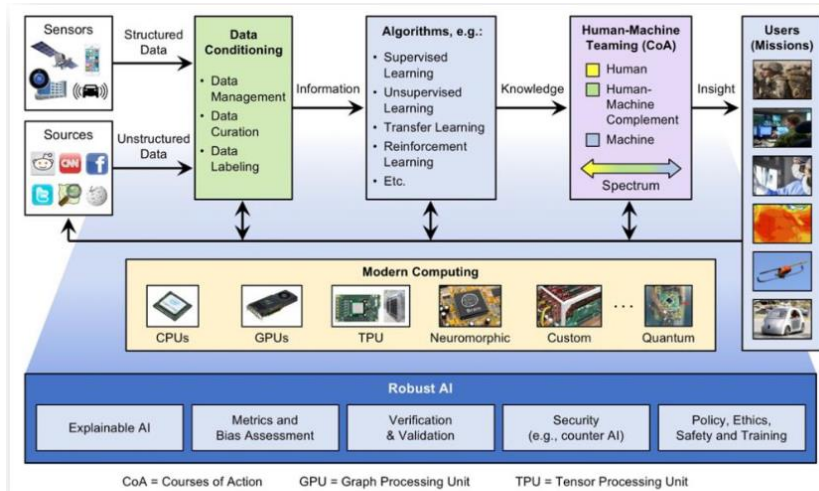
- Budget: 50,000 Euro from ESA, Internal Investment from Mission Control of 86,250 Euro
- Delayed start due to COVID-19 outbreak
- Mass storage event on OPS-SAT in September-October changed access and memory requirements for the experiment, requiring experiment re-design
- A number of successful versions of the experiment have been run on the ESA EM, producing results and the next release will be ready for FM deployment
- Suggest that Mission Control can amend the Final Report for ESA after the FM results are downlinked



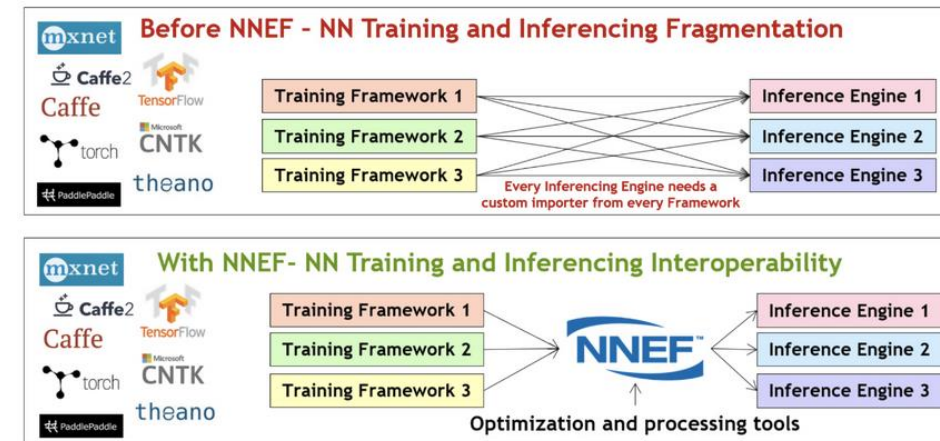
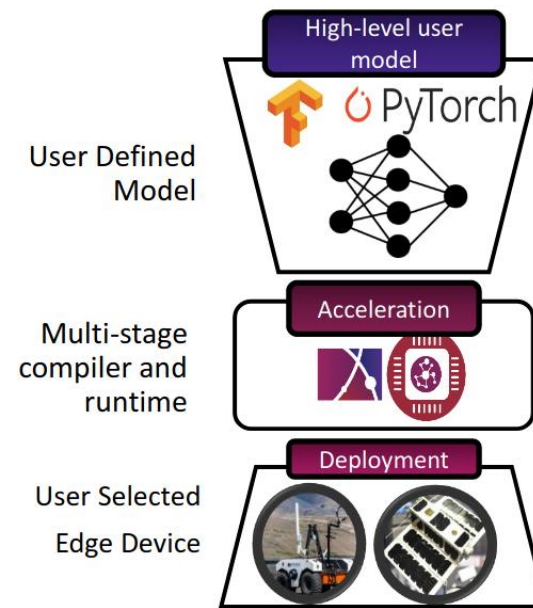
Technical Update

Introduction to Deep Learning Accelerator

- Mission Control has developed a product to deploy deep neural networks on spaceflight hardware
- This deep learning accelerator takes high-level torch or python models to FPGAs or low power devices

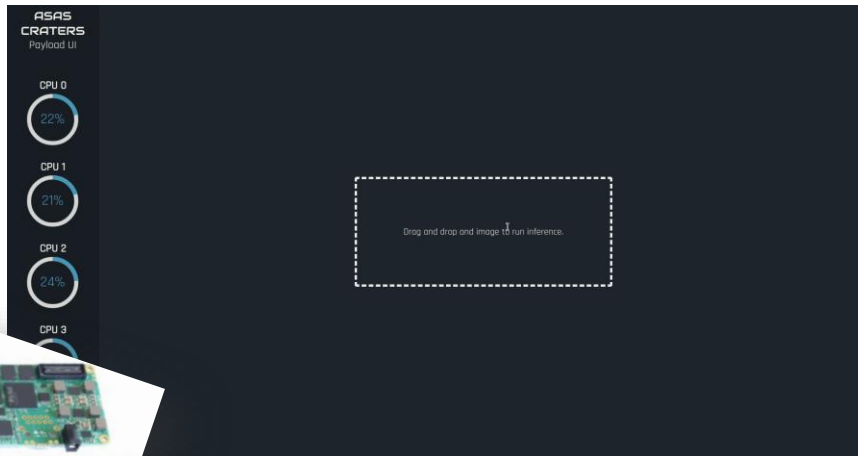
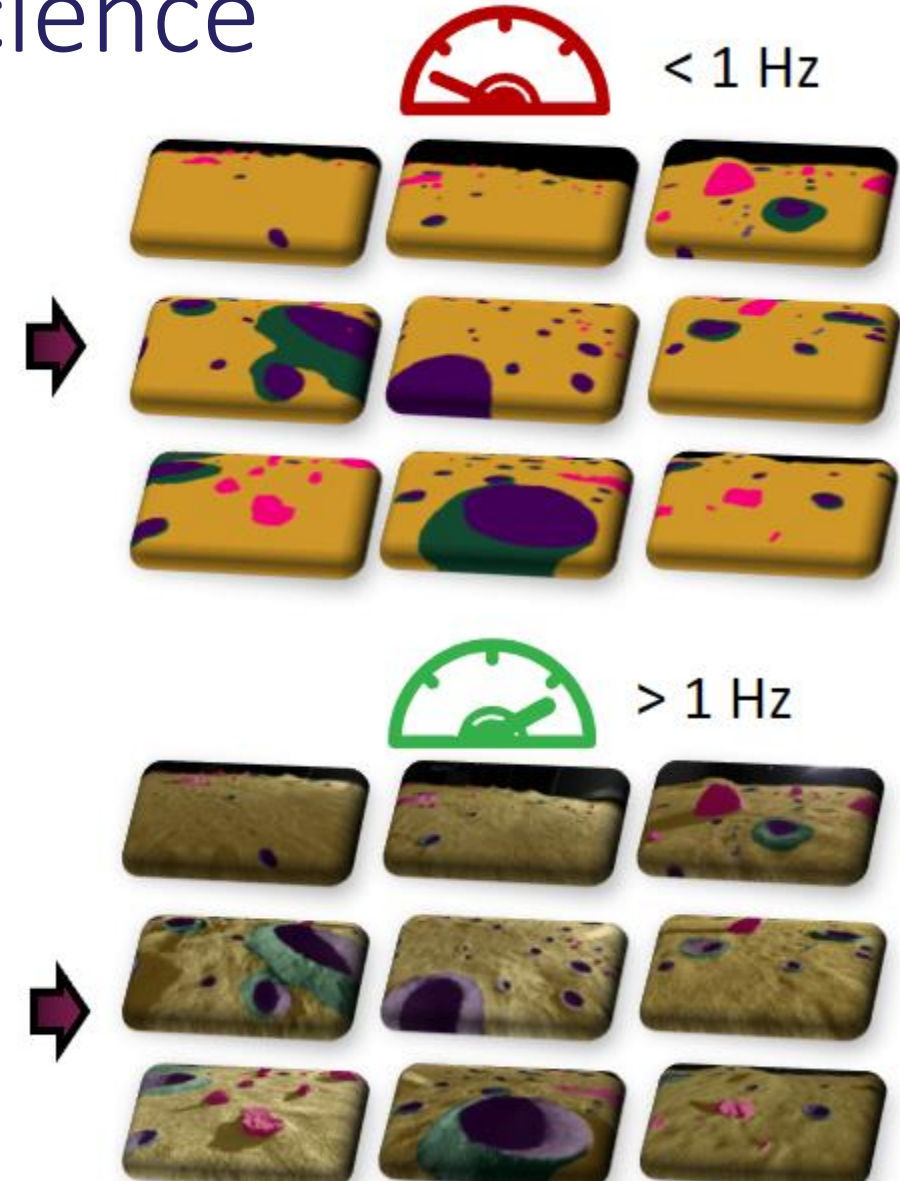
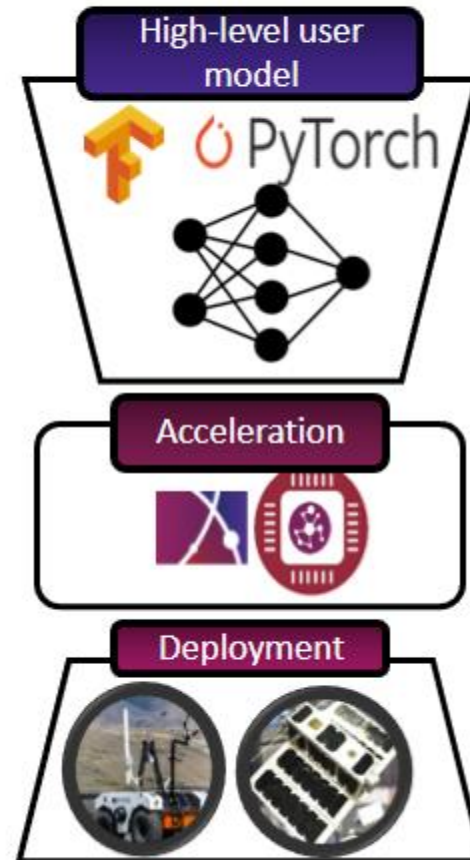
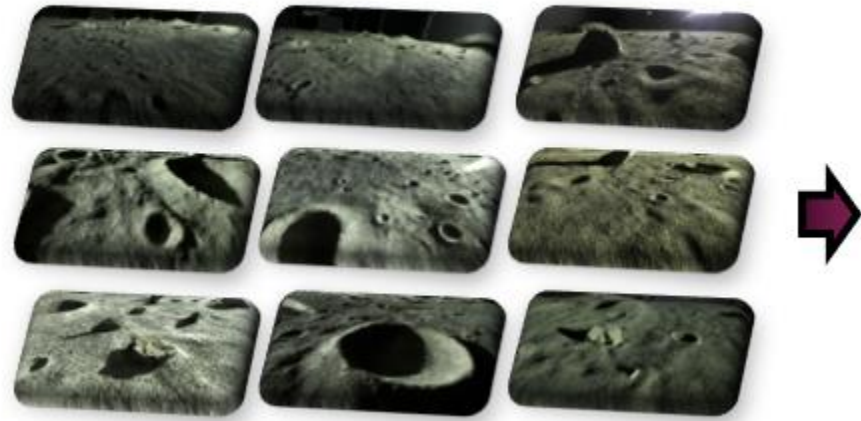


[Survey and Benchmarking of Machine Learning Accelerators](#)
[Towards the Use of Artificial Intelligence on the Edge in Space Systems: Challenges and Opportunities](#)



Credit: Khronos Group

Example - DLA for Planetary Science

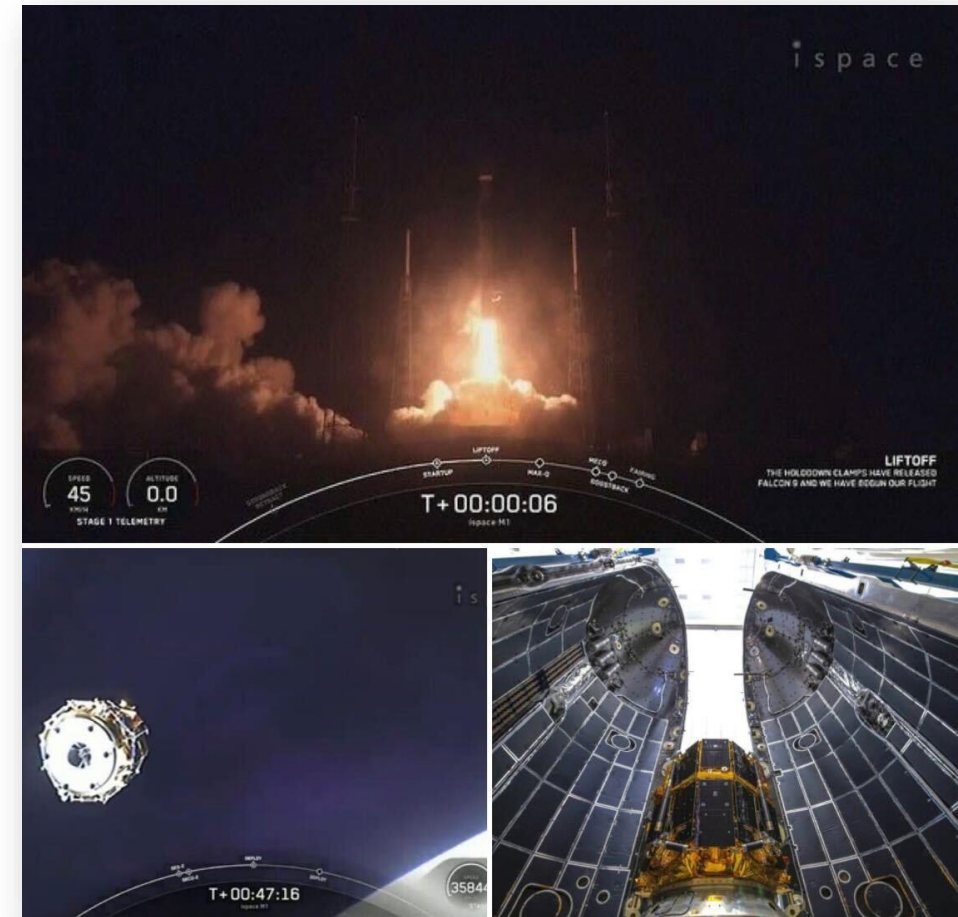


Introduction to the Deep Learning Accelerator

Launch on December 11th!



**First demonstration of
deep learning on the
Moon**



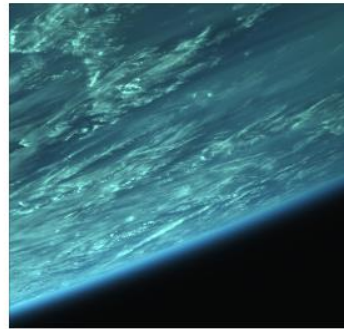
Credit: iSpace, SpaceX

SmartCam Overview

- SmartCam app performs image classification onboard OPS-SAT
 - Runs a pipeline with several classifiers
 - Autonomously determine which images to send
- Our focus: default TensorFlow Lite model
 - Convolutional neural network (CNN) based on the MobileNetV2 architecture
 - Pretrained image classifier to classify earth images into 3 classes
 - Earth, Edge, Bad



(a) Earth



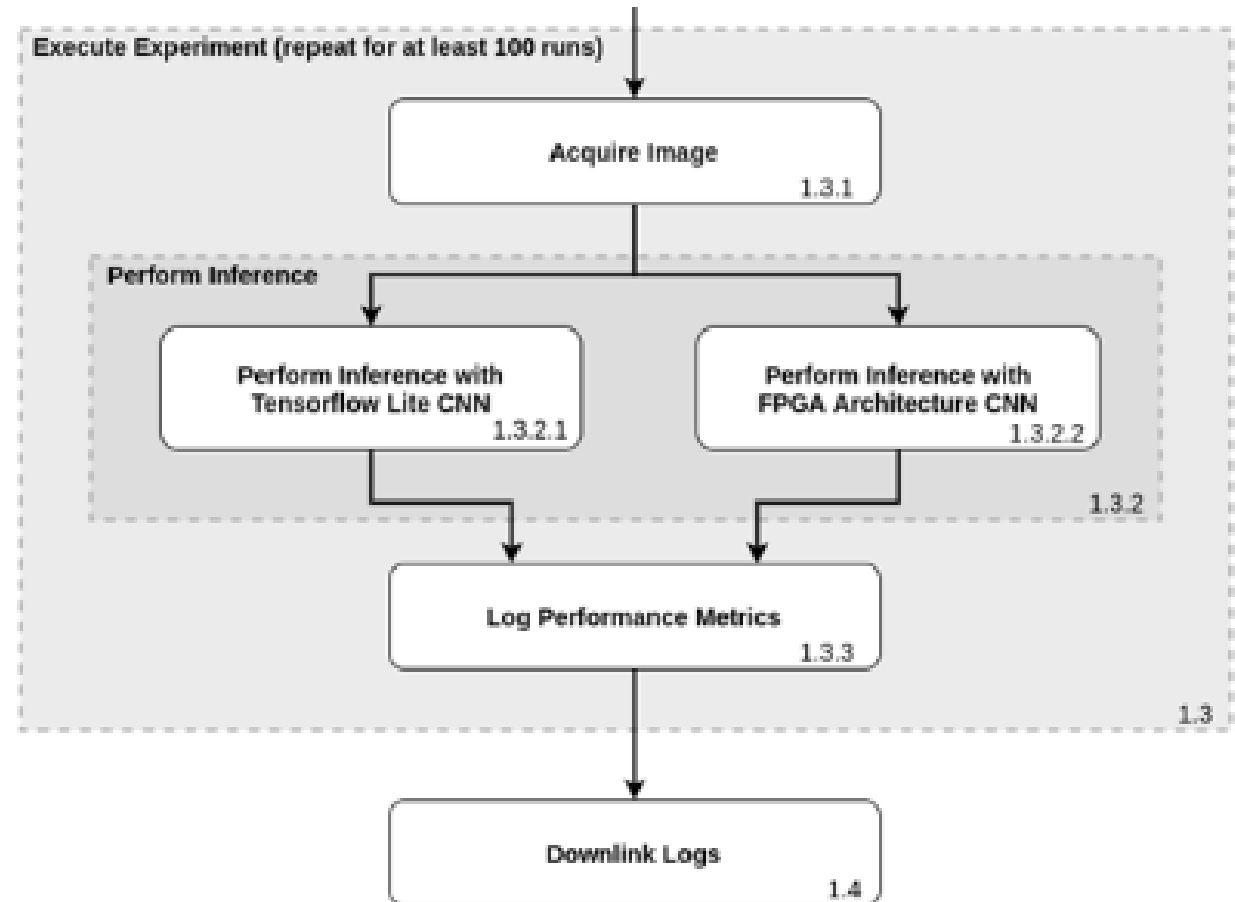
(b) Edge



(c) Bad

Experiment Overview

- Compare two inference approaches using the same CNN model
 - Original TensorFlow Lite implementation
 - DLP (Deep Learning Processor)
 - Uses NNEF industry standard for interoperability
 - Multi-stage compiler
 - Embedded runtime
 - Hybrid CPU + FPGA approach



Hardware Setup

- The team ultimately acquired three MitySOM-5CSX-H6-42A development boards, thank you to ESA for the loan of one of the boards
 - All three boards were used extensively with developers working in parallel to achieve our progress
- The MitySOMs were booted up and configured with a default image, as per the specification of the board manufacturer Critical Link



Initial FPGA Implementations

- Adder Experiment Structure:
 - Adder/subber implemented in FPGA logic
 - Device tree overlay makes device in FPGA available to CPU userspace
 - Program running on CPU gets inputs from user and writes them to device, then reads results back
- Developed test applications for:
 - Streaming data directly to DDR
 - Streaming data through a FIFO

```
root@mitysom-c5:~# ./adder_subber
The adder/subber has been reset. Here are its memory contents in...
Binary: 00000000 00000000 00000000 00000000 00000000 11111111 11111111 00000000
Decimal: 0 0 255 65280
Hexadecimal: 00 00 00 00 00 ff ff 00

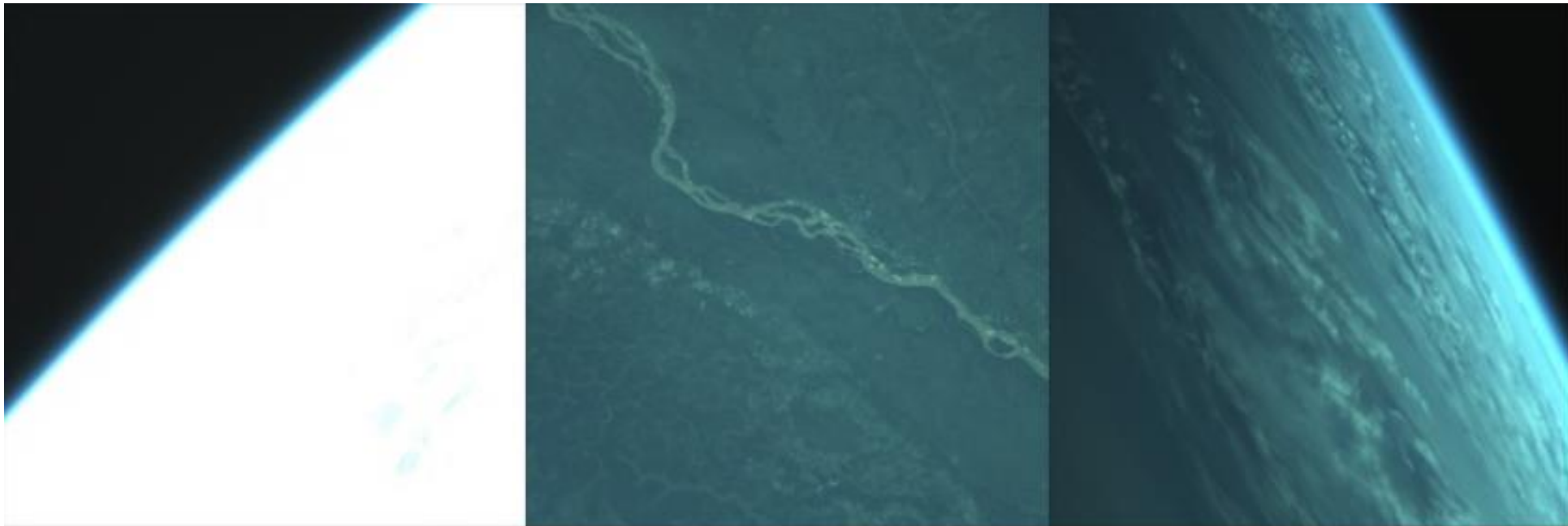
Input 4 numbers to add and subtract, in order and separated by newlines:
12
1
34
2

The inputs have been written to the adder/subber. Here are its memory contents in...
Binary: 00000000 00110001 11111111 11100111 00000000 11111111 11111111 00000000
Decimal: 49 -25 255 65280
Hexadecimal: 00 31 ff e7 00 ff ff 00

Here are the results obtained from the adder/subber:
12 + 1 + 34 + 2 = 49
12 - 1 - 34 - 2 = -25
root@mitysom-c5:~#
```

SmartCam Development – Test Dataset

- 100 images randomly sampled from labeled thumbnails on ESA's DMS
- 85 "Bad" images, 12 "Earth" images, 3 "Edge" images
- Modeled after distribution from the paper by Labrèche et al.



Bad

Earth

Edge

SmartCam Development – Metrics

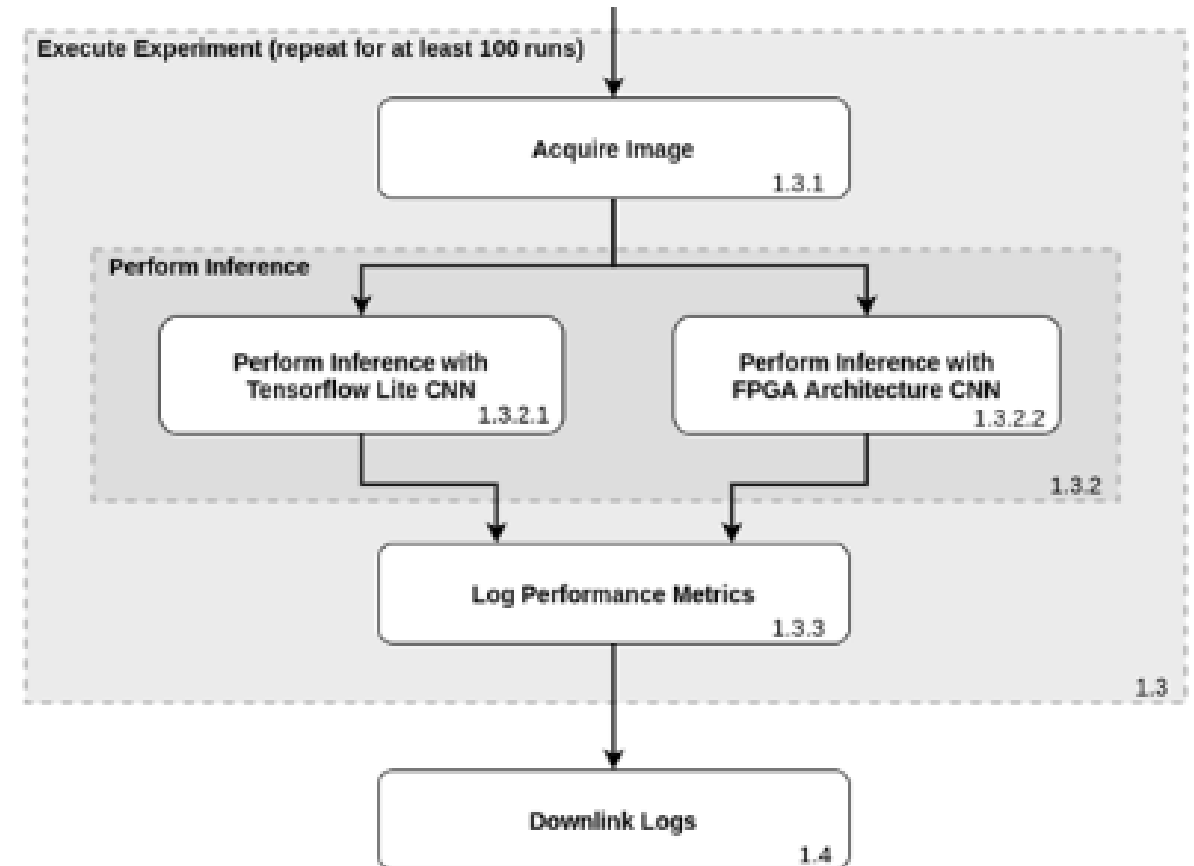
- Uses metrics used by Labrèche et al.:
 - Precision
 - Sensitivity
 - Specificity
 - F1 Score
- Additional model performance metrics:
 - Accuracy
- Most relevant, framework performance:
 - Average Elapsed Time
 - Average CPU Time



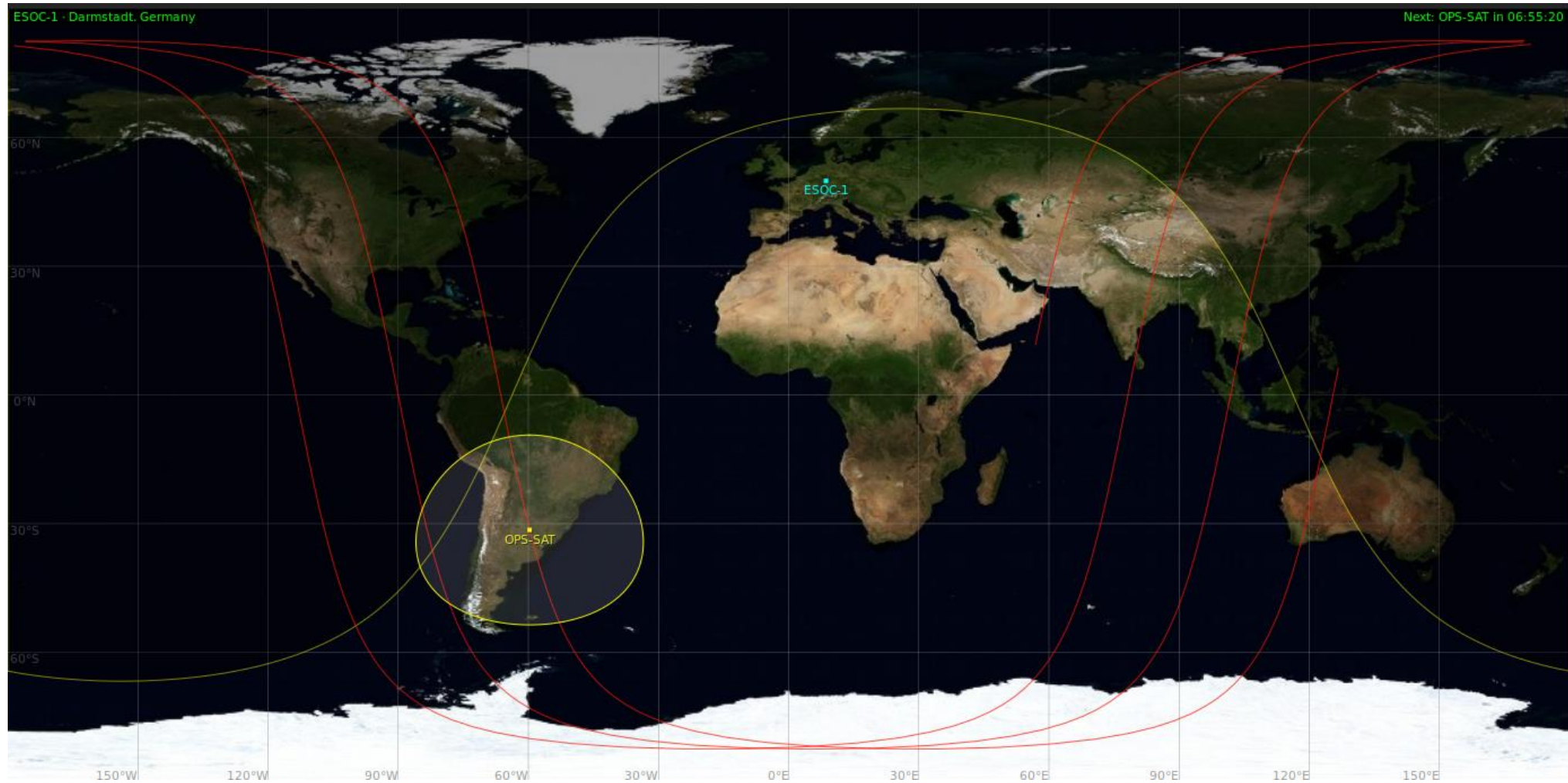
OPS-SAT SmartCam
i will neural network your earthies

Software Architecture

- After the initial app has been uplinked to OPS-SAT the experiment waits for a given date and time to start
- The process begins by iteratively capturing RAW images using the onboard camera utility
- Flatsat testing concluded that the process of capturing an image, converting it from RAW to PNG, resizing and saving takes around 20 seconds.



Orbital Path Planning



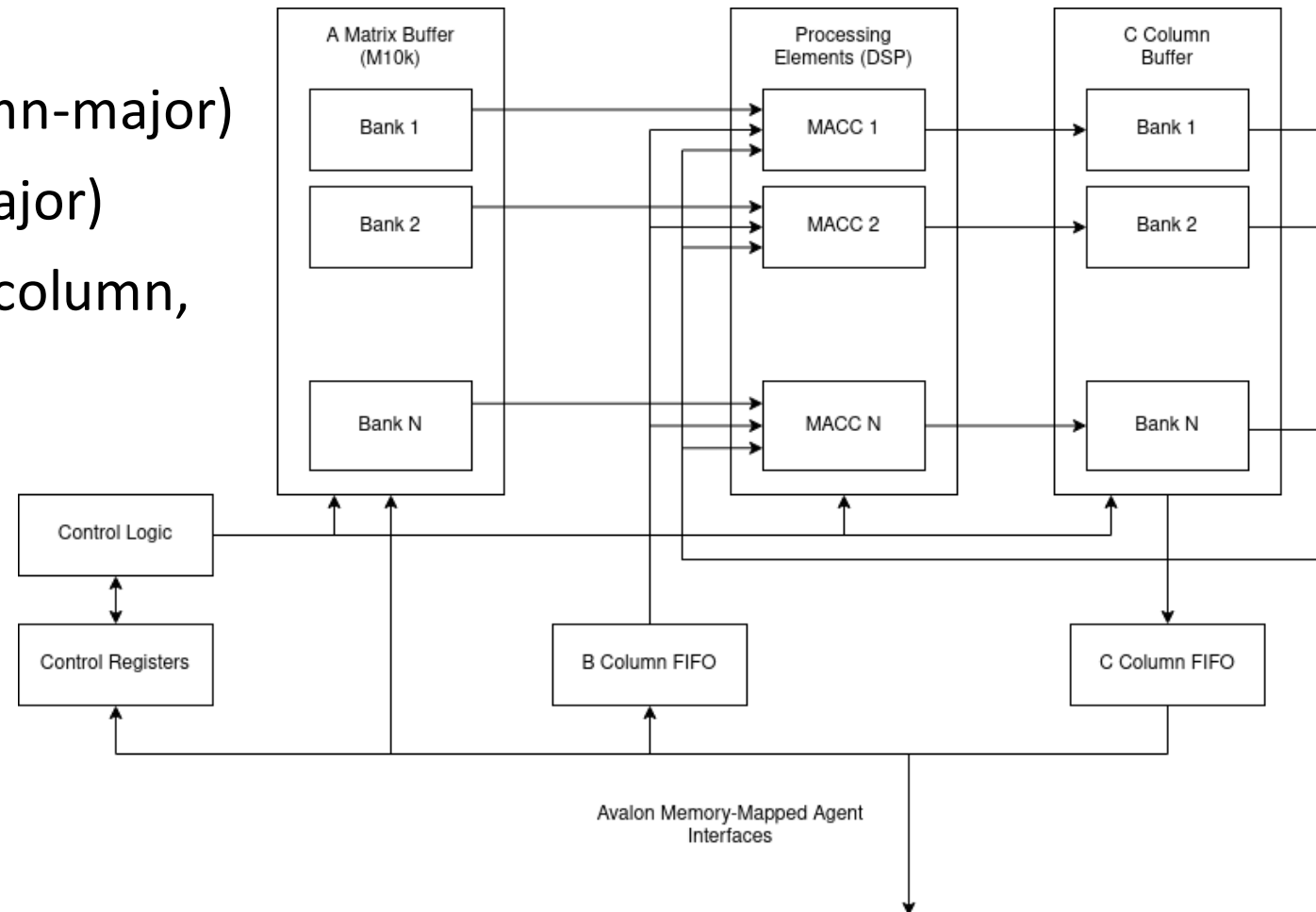
FPGA Component

- HDL vs High-Level Synthesis (HLS)?
 - Familiar toolchain vs. deeper toolchain
 - Lower-level vs. higher level implementation
 - More code vs. less code
- HLS matrix multiplier implementation in <100 lines of C++
- Quick to get a working implementation
 - After setting up toolchain
- Unpredictable effects on performance
- Optimization ceiling
- Was it the right choice? Hard to say.



FPGA Component – Design

- $A * B = C$
- Store A in on-chip memory (column-major)
- Stream elements of B (column-major)
- Multiply each B element by each column, accumulate in C column buffer
 - This can be parallel
- Stream C buffer back to DDR
- Repeat for each column of B



Direct Memory Access

- In order to offload the processor from having to move around data, the team attempted to use a Modular Scatter-Gather Direct Memory Access (mSGDMA)
- This component performs data movement utilizing pre-set instructions called descriptors, which can be called in parallel
- MSGDMA would allow multiplication and data transfer to happen in parallel, thus optimizing FPGA performance.
- Not yet fully integrated into current release

Experiment Versions

- Test versions
 - 0.1: FPGA adder/subber example
- CPU-based
 - 0.2:
 - Initial version of the complete experiment
 - Unable to run it on the EM due to high memory consumption
 - 0.3:
 - Memory consumption improvements
 - Still unable to run it on the EM due to high memory consumption
 - 0.4:
 - First version that executed successfully on the EM
 - Captured images from the EM camera
 - 0.5:
 - Same as version 0.4, but using images from test dataset
 - Test dataset and fake camera app embedded in the installation package

Experiment Versions

- FPGA-based
 - 1.0:
 - First version using the FPGA to accelerate inference operations
 - Continues to use test dataset
 - FPGA operation validated on MitySOM boards
 - Submitted to ESA for EM testing
 - 1.1:
 - Same as version 1.0, but using images from the camera
 - To be submitted to ESA after 1.0 is validated on EM
 - Final version intended to run on the FM

MitySOM Testing and Evaluation

- Version 0.5
 - CPU implementation of all inference operations
 - Using test dataset embedded in the application for validation on EM and MitySOM

Framework	Accuracy ↑	Precision ↑	Sensitivity ↑	Specificity ↑	F1 Score ↑	Avg. Elapsed Time (ms) ↓	Avg. CPU Time (ms) ↓
TensorFlow Lite	100%	1.0	1.0	1.0	1.0	920.08	918.93
DLP (CPU)	100%	1.0	1.0	1.0	1.0	5130.48	5125.03

MitySOM Testing and Evaluation

- Version 1.0
 - FPGA implementation of certain inference operations
 - Using test dataset embedded in the application for validation on EM and MitySOM

Framework	Accuracy ↑	Precision ↑	Sensitivity ↑	Specificity ↑	F1 Score ↑	Avg. Elapsed Time (ms) ↓	Avg. CPU Time (ms) ↓
TensorFlow Lite	100%	1.0	1.0	1.0	1.0	920.28	919.32
DLP (FPGA)	100%	1.0	1.0	1.0	1.0	5422.03	5415.85

Engineering Model Testing and Evaluation

- Version 0.5
 - CPU implementation of all inference operations
 - Using test dataset embedded in the application for validation on EM and MitySOM

Framework	Accuracy ↑	Precision ↑	Sensitivity ↑	Specificity ↑	F1 Score ↑	Avg. Elapsed Time (ms) ↓	Avg. CPU Time (ms) ↓
TensorFlow Lite	100%	1.0	1.0	1.0	1.0	1599.18	1524.07
DLP (CPU)	100%	1.0	1.0	1.0	1.0	8644.13	8260.88



Conclusions

Conclusions

- Mission Control developed a hybridized version of the ESA OP-SAT SmartCam experiment using its Deep Learning Accelerator technology
- A series of tests of increasing complexity was run on MitySOM development boards and the ESA EM, starting from an FPGA adder experiment and building towards the fully hybridized experiment, which is now staged to run on the EM with a small change required to run Version 1.1 on the FM
- Further implementation of Direct Access Memory shows potential for increasing the performance of the Mission Control implementation and eventually surpassing the TFLite CPU implementation
- Moving intensive matrix multiplications required for CNNs onto the FPGA could free up the CPU for other aspects of mission operations

Next Steps

- Delivery of all deliverables specified in the contract today
 - MCSS.2112 OPS-SAT Executive Summary
 - MCSS.2113 OPS-SAT Final Report
 - MCSS.2114 OPS-SAT Final Review Presentation (this deck)
 - MCSS.2115 OPS-SAT Activity Illustration (x2)
 - MCSS.2116 OPS-SAT 3-Minute Video
 - And meeting minutes from final review
- Coordination with ESA on the running of Version 1.0 on the EM and Version 1.1 on the FM, updating deliverables as required
- Mission Control plans to continue improving its DLP product stack, using DMA and working to surpass the TFLite runtime
- Thank you to everyone at ESA and on the OPS-SAT team for your coordination and assistance throughout the 2022!