Deep Learning at the Edge of Space

Rahul Vishnoi[a], Dr. Mary Bowden[b], Dr. David L. Akin[c]

Abstract

Balloon payloads are a cost effective and reliable method for testing hardware and software in spacelike environments without the exorbitant relative cost of rockets. At the University of Maryland College Park, the Balloon Payload Program provides this opportunity to students, enabling them to test student experiments in challenging near space environments. The Super Complicated Ai Mission Payload (SCAMP) leverages this opportunity to develop a payload capable of flying neural networks in these environments on an extremely low-SWaP platform that is both exceptionally affordable and reliable. This is accomplished by utilizing NASA's core Flight Software (cFS) and commercial off the shelf (COTS) component-based design. As a byproduct, this payload also demonstrates an approach that enables the integration of neural networks into NASA's cFS using Google's Coral Tensor Processing Units (TPUs). Furthermore, the payload also proves the flight worthiness of the Coral TPUs in harsh space-like environments. Finally, this payload provides a low barrier of entry for students interested in experimenting with AI, NASA cFS and software design, further enriching students' understanding of flight software, embedded system design and programming.

Neural Processing Units | Deep Learning | NASA core Flight System | Artificial Intelligence | Machine Learning | High Altitude Ballooning | Ai Acceleration | Tensor Processing Unit | Flight Software | Embedded Software

---

1. Introduction

The Super Complicated Ai Mission Payload (SCAMP) originally started out as a weekend project to test if it was possible to put a neural network into NASA's cFS. The project's objective was to create a simple balloon payload that was cheap and easy to build. To that end, SCAMP was developed primarily from components found in the trash, around the University of Maryland College Park and eBay. While not using the most flight worthy hardware, or even the most flight qualified, SCAMP was able to demonstrate deep learning at the edge of space for a price tag of about 80 dollars. Through this cheap payload, it was also possible to lower the barrier of entry for others who wanted to run more complex experiments at altitude with neural networks.

2. SCAMP (Super Complicated Ai Mission Payload) v1 Design

Keeping costs low and reducing complexity of the payload resulted in the hardware itself consisted of the cheapest COTS components available. Consequentially, none of these components were rated for the environments that they would be operating in but through the flights, they proved to work just fine. The initial configuration of the payload consisted of a Coral Dev Board which costed about 48 dollars through eBay at the time of purchase; a USB Camera unit that costed about

[a]M.S. Student, Department of Computer Science, 1309 Kim Engineering Building, rvishnoi439@gmail.com
[b]Senior Lecturer, Department of Aerospace Engineering, 3181 Glenn L. Martin Hall
[c]Professor of Aerospace Engineering and Director, Space Systems Laboratory, Department of Aerospace Engineering, 3181 Glenn L. Martin Hall

15 dollars through Amazon at the time of purchase; and finally, a phone travel power bank that was acquired through a career fair but can retail for about 15 dollars through Amazon. This kept the budget of the payload well below 100 dollars at the time of creation.

The initial configuration is shown below in figure 1. This configuration had the Coral Dev Board placed on the lid of the payload to simplify the wiring and maintenance of the payload after

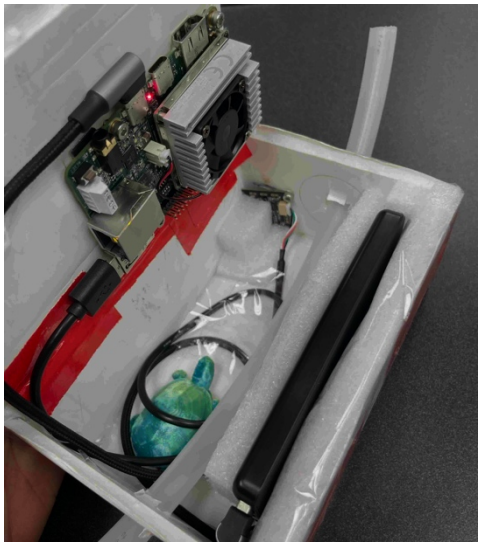

Figure 1: Initial payload hardware configuration

Figure 2: Current payload hardware configuration post NS-129 failure

flights and during debugging. This decision, while optimal for access to the board and maintenance of the payload, resulted in a hardware failure on SCAMP's second flight (NS-129). The source of the problem was that the Coral Dev Board is two boards with the SoC (System on Chip) component being attached to a breakout board with all the IO (input output) peripherals. During the second flight, on the launch pad, the payload was tapped aggressively causing the pins securing the SoC to detach from the breakout board completely. This disconnection occurred on the pins associated with the USB port connecting the SoC to the Camera, crashing the application responsible for inference. Post NS-129, the internals of the payload were rearranged to ensure that the board was more stable, resulting in the layout shown in Figure 2 above. While less maintainable, this configuration provides the board more safety thus making the payload hardware more reliable.

Due to the COTS nature of SCAMP, the electrical interfaces were rather simple. The entire payload consisted primarily of 2 cables. One cable connected the Coral Dev Board with the power bank to provide power to the whole payload while another USB cable provided data from the camera to the Coral Dev Board. The entire system had an estimated runtime of about 5 hours at peak load when the power bank was at full charge.

The flight software onboard SCAMP utilizes NASA's core Flight System (cFS) as its primary flight software framework. NASA cFS is a TRL 9 flight software framework provided by NASA Goddard for free due to its open-source nature. cFS comes with a multitude of open-source applications built to run on top of cFS provided which helps enable rapid prototyping and testing of flight systems with little downtime. Furthermore, cFS provides a soft real-time system to help ensure timing requirements met and that tasks onboard the payload are sufficiently scheduled based on explicitly defined priorities.

SCAMP's software architecture leverages several of cFS' components to create the architecture described in Figure 3. Outside from the core Flight Executive services provided by cFS, SCAMP utilizes three of cFS' open-sourced applications - Stored Command, Scheduler and

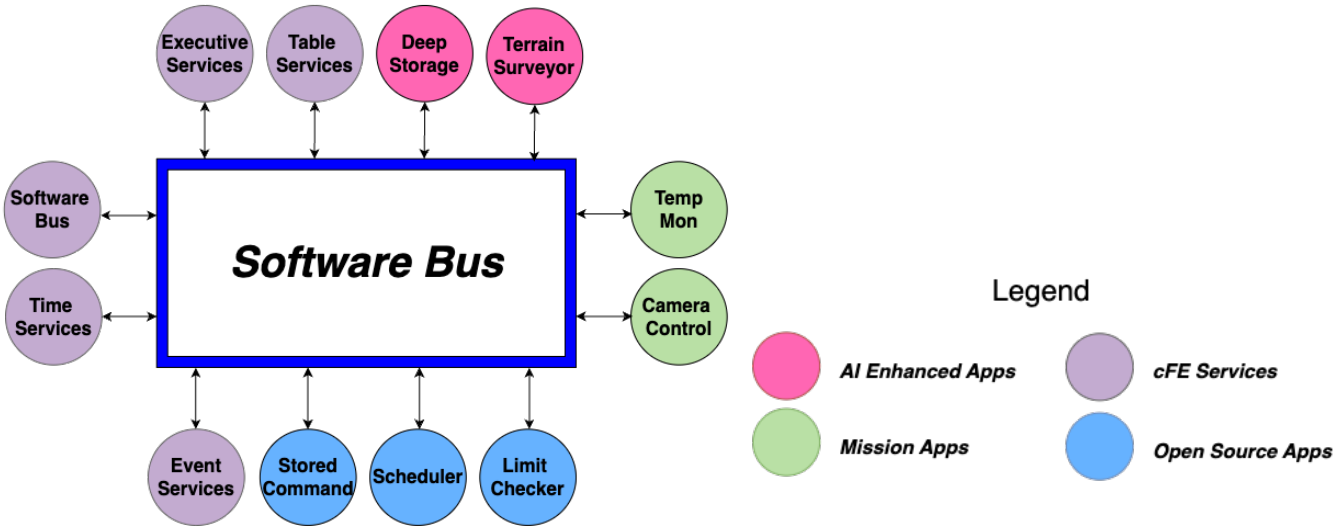

Figure 3: SCAMP cFS Software Architecture

Limit Checker - to provide control over the main system. Stored Command and Limit Checker exist to ensure that the payload does not overheat or throttle by ensuring temperatures provided by the Temp Mon application (a custom SCAMP application) are within nominal operation ranges. Additionally, Stored Command is also utilized to run relative time sequence (RTS) commands to monitor payload utilization through cFS' built-in performance monitor. Finally, the Camera Control Application gathers images from the payload camera and saves them to the file system in a critical data store while dumping the address to the image on the software bus.

3. SCAMP v1 Networks & AI Enhanced Applications

Alongside the standard applications SCAMP consists of two AI enhanced applications that run quantized neural networks on the onboard tensor processing unit (TPU). On the Coral Dev Board, the TPU is integrated with the CPU on the SoC portion of the Coral Dev Board and commands through a PCIE bus that connects the two processing units. To ensure that the networks can run on the resource constrained TPU, the models must be quantized accordingly. Quantization adjusts the weights within the model by encoding them within unsigned integers rather than representing the weights as 64- or 32-bit floating values.

Pytorch was used to develop and train the networks utilized on the payload due to its low learning curve, ease of use, and open-source support. To make it possible to transfer the trained models to be deployable, a pipeline was required to encode the weights in a quantized state. Since the Coral TPU was designed by Google, they utilized the TensorFlow Lite representation for saving quantized weights for the TPU. To achieve this, Google create a tool called torch-edge-ai[1,2]. This utility makes it possible to take a Pytorch model and convert it to a TensorFlow Edge Lite model. This model is then converted to a final TPU representation through the Coral TPU Compiler which takes a quantized TensorFlow Lite model and converts it into a binary blob that contains the instructions for the network's inference that can then be read by the Coral TPU. With this workflow, developing networks for SCAMP become nearly plug and play.

The first model that was developed through this pipeline was the Terrain Surveyor network. The model utilizes a UNet[3] architecture to segment ground images to identify what the terrain below the payload is. The data that was used for this training this model was provided through an opensource project on Kaggle that aggregated images across several aerial image datasets[4]. The

dataset came with 7 normalized groups that the segmentation model understands: buildings, land, roads, vegetation, water, objects and undefined. This network was loaded into the Terrain Surveyor application which waits on images from the Camera Control application and then runs the inference step on the images. The fastest this app inferred on an image, based on the flight data, was around 5 milliseconds. This meant it was possible to infer on images at about 30 frames per second onboard the payload on the NS-128 flight.

The second application that is enhanced by AI is the Deep Storage Application. The idea behind this application is to take the image data and compress the images utilizing an autoencoder[5]. The way the autoencoder is trained is rather different than the standard training paradigm since dealing with quantization can affect the autoencoder's understanding of a latent space unpredictably. As such the encoder is a pretrained quantized Inception v4 model[6] without its final SoftMax layer. The decoder is then trained and deployed on the ground without any quantization. The training data used for building this network was the image data from NS-128 and NS-130. As such the first flight that utilized this model, and application will be NS-131.

4. SCAMP v1 Flight Data

SCAMP's maiden flight was Near Space 128 (NS-128). This flight has a very simple cFS application setup that consisted of the Terrain Surveyor Application, Scheduler, Camera Control alongside all the built in cFE applications. SCAMP's payload hardware layout, at the time of the flight, was in the orientation shown in Figure 1. As far as data goes, Figure 4 shows some of the

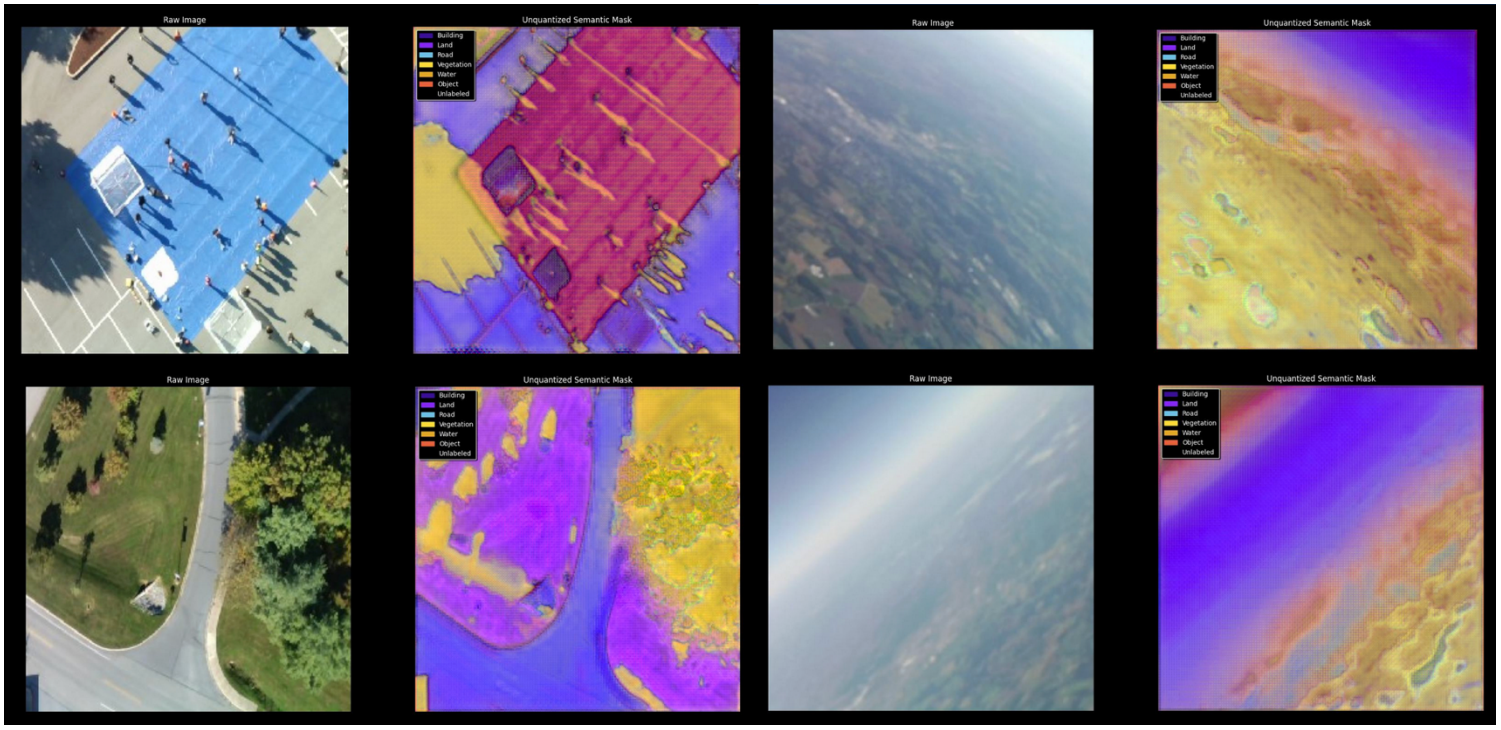

Figure 4: SCAMP NS-128 Flight Data

images and inferences conducted by the Terrain Surveyor Application during flight. In these images there are quite a lot of poor hallucinations visible. For example, in the two images on the right, the Terrain Surveyor model describes the horizon and space as a building. Throughout the data the vegetation group always appeared to be consistently correct, except for the top left

example in Figure 4 that describes shadows as vegetation. The bottom left example in Figure 4 is one of the best inferences that was achieved by the Terrain Surveyor Application.

Almost all the faults for these abnormally incorrect behaviors in the model can be pinned on the quality of the dataset utilized for training the network. The dataset consisted of an aggregation of multiple different types of aerial segmentation images ranging from drone images to satellite images. As such the model may have over generalized to some degree and failed to really understand some of its predictions. Furthermore, none of the images had any shots with space which explains why the model would predict it being a building. Finally, the model was trained on about 400 images. There were not enough images in the dataset to make the model behave better and as a result the outcomes from NS-128 were almost predestined.

After NS-128, the Temp Monitor Application was added into the software stack to provide more data and information on the payload's health throughout the flight. Tragically, however the Near Space 129 (NS-129) flight was a failure due to the hardware. The problem with this launch revolved around the placement of the Coral Dev Board inside of SCAMP. After this launch the hardware was redesigned and ended up in the configuration shown in Figure 2. While the Camera Control Application and Terrain Surveyor did not collect anything, the Temperature Monitor Application was still able to function. Figure 5 displays the temperatures of the Coral Dev Board.

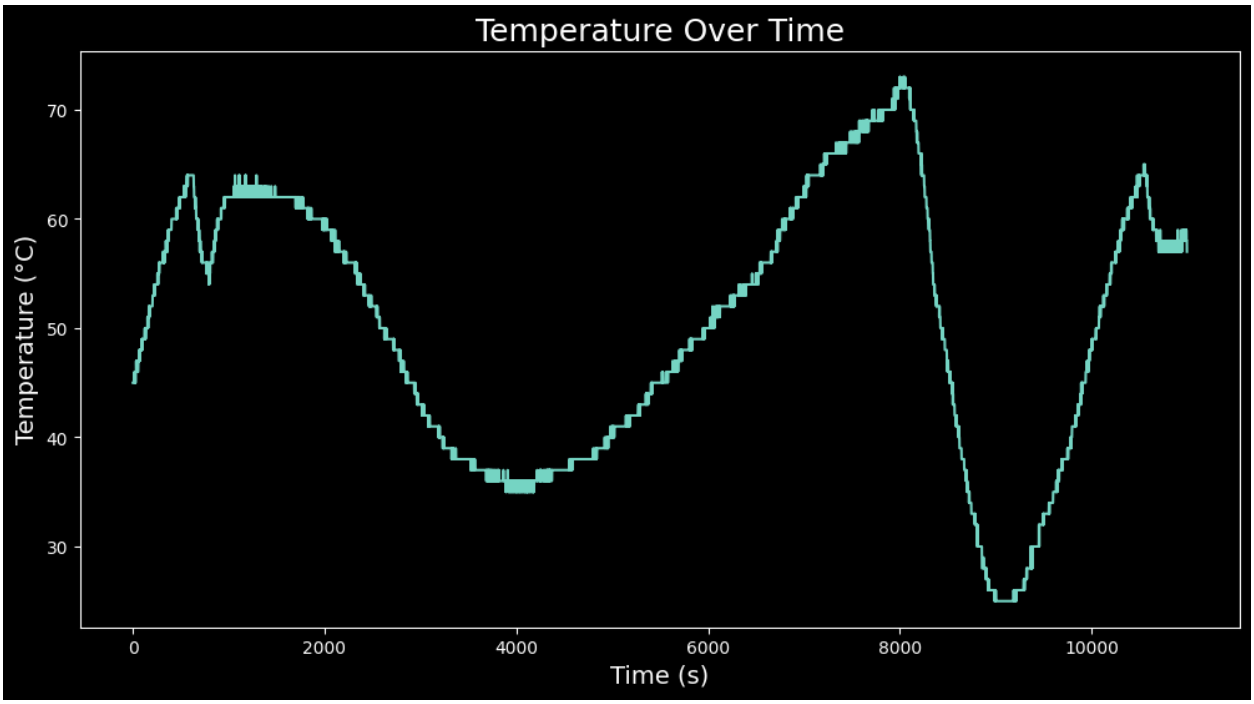

Figure 5: SCAMP Payload Temp during NS-129

What is critical here is that the temperature range remained within the ranges specified by Google for maintaining the health of the Coral Dev Board (-40° C to 85° C).

After changing the hardware, SCAMP was ready to fly on NS-130. Between NS-129 and NS-130 there were no software changes aside from code cleanup and documentation that was left out due to tight deadlines. Unlike NS-129 however, NS-130 was a success. The most interesting piece of data received from this flight was understanding if model inference time was affected by

temperature during the flight. This hypothesis was proven false by Figure 6. Asides from this

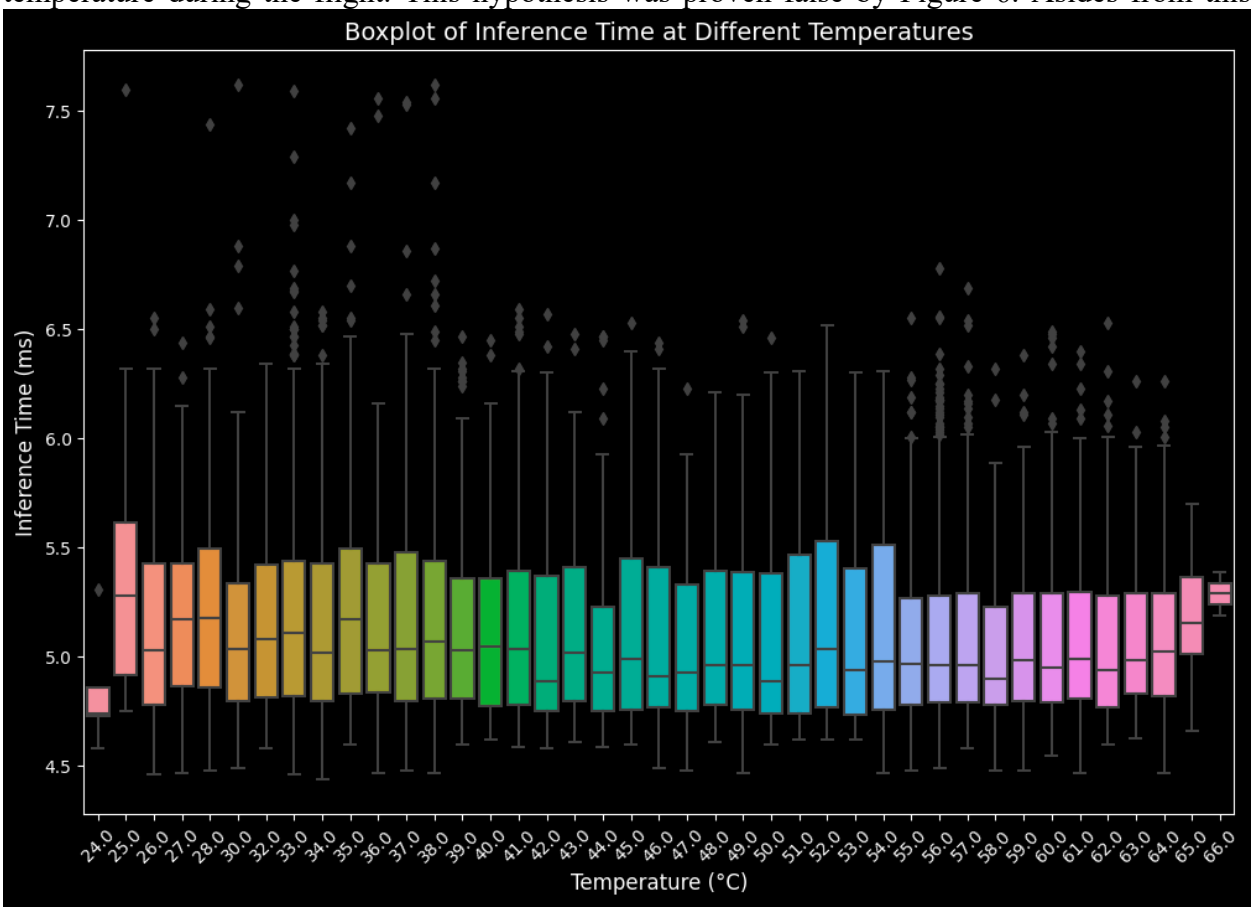

Figure 6: SCAMP NS-130 Temp vs Inference time plot

finding, NS-130 did not fly a new model for Terrain Surveyor and as such the same hallucinations present in NS-128 were also prevalent in NS-130.

Post NS-130 it was clear that data describing each threads runtime and status was required and the idea to stress test the TPU with multiple networks was brought up. As such between NS-130 and NS-131 Stored Command, Limit Checker and Deep Storage were added into the flight software.

5. Conclusions & Future Work

The next step with the project is to keep pushing these TPUs to their limits. NS-131 and NS-132 shall provide a good understanding of how far these TPUs can be pushed.

Outside of SCAMP v1, SCAMP v2 consists of three Coral USB TPUs that can split the load between six neural networks and applications. The software architecture in Figure 7 depicts the planned applications that are to be part of this payload. The whole intent behind this is to try and create a more realistic software stack that can expected on a real NASA mission to see how useful AI enhanced applications are and how performant they can be. Going through the applications themselves, Deep Storage and Terrain Surveyor are carried over from SCAMP v1 and not disturbed at all given their flight heritage. Deep Navigation or Deep Nav is an application that utilizes the Sensor Control Application to collect environmental data from a sub-payload that is then used to estimate the position of the payload relative to the launch site. The raw sensor data

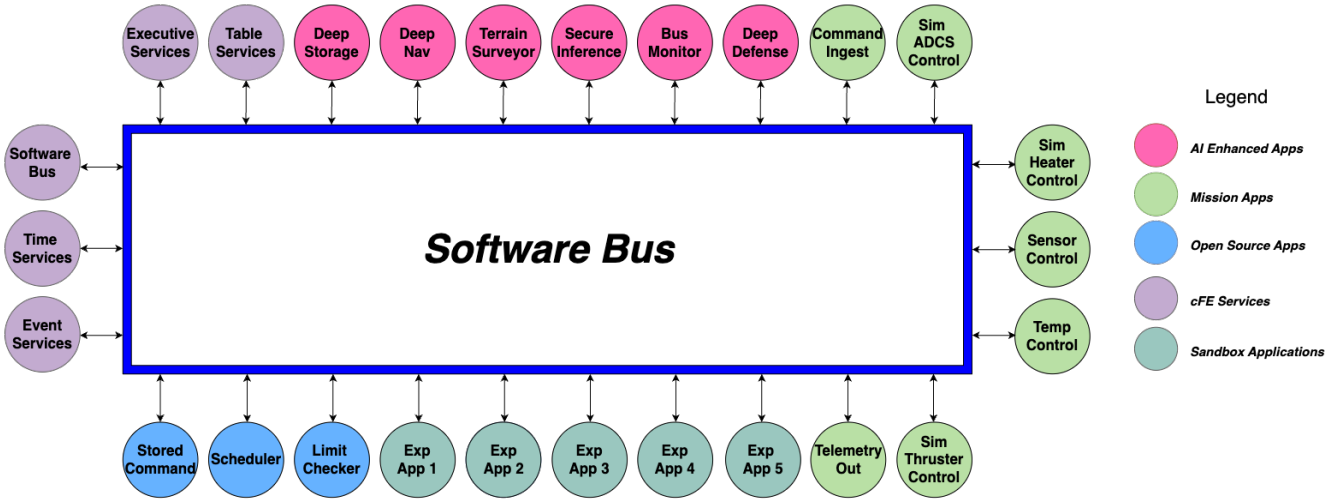

Figure 7: SCAMP v2 Software Architecture

provided from the Sensor Control package at a rate of 1Hz consists of raw magnetometer readings, accelerometer readings, gyroscope readings, pressure, temperature, and a calculated altitude given the pressure and temperature data. Unlike the other networks in Terrain Surveyor and Deep Storage, Deep Nav utilizes Long Short-Term Memory (LSTM) blocks[7] within its model.

Deep Defense also utilizes LSTMs but instead of finding the position of the payload, it monitors the health of all the apps onboard and ensures that none of the apps are corrupted. To test this idea, SCAMP v2 has 5 Exp Apps that are barebone applications that will randomly appear as corrupted and will try and send bad data to all the Simulation Apps onboard such as Sim Thruster Control. The Bus Monitor application is a lot like the Deep Storage Application except rather than compressing data, it utilizes auto encoder to study the structure of data through its reconstruction loss. Finally, the Secure Inference Application is an application that performs inference on encrypted data and output its findings with the same encryption pattern. This payload is fully developed and programmed, if there is an opportunity to fly it before graduation that would be cool but otherwise it makes for an interest ground test bed environment.

Overall, The Super Complicated Ai Mission Payload proved that cutting-edge technology like neural networks can be integrated into spaceflight systems on a tight budget. Despite using salvaged and third hand parts, SCAMP successfully ran several deep learning networks in near-space conditions, showcasing both the potential of NASA's cFS for AI integration and the power of low-cost, grassroots engineering. SCAMP not only met its goal as a proof-of-concept but also opened the door for future accessible, affordable, and innovative high-altitude AI research for other students.

Acknowledgments:

The author would like to acknowledge Charles Hanner, Melissa Buys, Daniil Gribok, and Meredith Embrey for their support with the initial development of the hardware designs and mentorship with the initial payload creation. The author would also like to acknowledge Romeo Perlstein, JJ Kuznetsov and Akemi Takeuchi for their support and mentorship with construction and assembly of the payload hardware. Finally, the author would like to acknowledge Zoe Roy, Cody Deyarmin, Sara Garcia-Beech, Eric Pollack, Eshwar Singh, Beth Geist, Alan Cudmore, Dan Berry, and Adrian Rodriquez from NASA Goddard for their mentorship, guidance and support with cFS.

Acronyms:

SCAMP	Super Complicated Ai Mission Payload
SWaP	Size Weight and Power
cFS	Core Flight System
AI	Artificial Intelligence
COTS	Commercial of the Shelf
SoC	System on Chip
USB	Universal Serial Bus
NASA	National Aeronautical Space Association
TRL	Technology Readiness Level
TPU	Tensor Processing Unit
CPU	Computational Processing Unit
NS-x	Near Space x (flight number)
LSTM	Long Short Term Memory

References:

1 Brick, Cormac, et al. “Ai Edge Torch: High Performance Inference of Pytorch Models on Mobile Devices.” Google Developers Blog, 14 May 2024, developers.googleblog.com/en/ai-edge-torch-high-performance-inference-of-pytorch-models-on-mobile-devices/.

2 “Convert PyTorch models to LiteRT” Google, Google, 2024, ai.google.dev/edge/litert/models/convert_pytorch.

3 Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18. Springer international publishing, 2015.

4 Rios, Alexander Daniel, “Aerial Image Segmentation” Github, https://github.com/aletbm/Aerial_Image_Segmentation/blob/8b980e13b27517c566c3635a9f2a571d2315b4ba/aerial-image-segmentagtion.ipynb

5 Bank, Dor, Noam Koenigstein, and Raja Giryes. "Autoencoders." Machine learning for data science handbook: data mining and knowledge discovery handbook (2023): 353-374.

6 Szegedy, Christian, et al. "Inception-v4, inception-resnet and the impact of residual connections on learning." Proceedings of the AAAI conference on artificial intelligence. Vol. 31. No. 1. 2017.

7 Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.