HALO: High Autonomous Low-SWaP Operations

Team Members:

- Sloan Hatter shatter2022@my.fit.edu
- Blake Gisclair bgisclair2022@my.fit.edu

Faculty Advisor: Dr. Ryan T White rwhite@fit.edu

Client: Dr. Ryan T White rwhite@fit.edu

Progress Matrix of Milestone 4

Task	Completion %	Sloan	To Do
Retrain 32-bit model on satellite dataset	100%	Retrained the full 32-bit model on satellite specific dataset from advisor	None
Implement FP16 engine from FP32 ONNX model	100%	Compile the FP32 ONNX model into a TensorRT engine using FP16	None
Implement INT8 engine	90%	Build an INT8 engine using the TensorRT framework from the Jetson SDK	None
Write an inference script for the INT8 model	100%	Write a Python inference script for the INT8 engine	None
Compute/record metrics for 8-bit model	100%	Get inference metrics from the INT8	None

Discussion of Accomplished Tasks for Milestone 4:

- **Retrain 32-bit model on Satellite Dataset**
  - My advisor provided me with a dataset containing various images of satellites in order to retrain the 32-bit model on more specific data. This model was then formatted into an ONNX file; an ONNX file is a standardized, open-source format used to store models, allowing a model to be trained in one environment and be deployed and executed in a completely different environment or hardware platform.
- **Implement FP16 engine from FP32 Model**
  - The FP32 model was compiled into a TensorRT engine using half-precision floating point (FP16). This was done using TensorRT's trtexec tool and a fixed input shape of 1x3x800x800, which produced the FP16 model. TensorRT is NVIDIA's inference optimization and runtime system for deploying neural

networks on NVIDIA GPUs. It takes a trained model and produces an optimized, device-specific executable called an engine. That engine inserts optimizations so inference runs faster and more efficiently than a generic framework runtime. The trtexec command-line tool builds an engine from a given model; for this project, it built the FP16 engine from the FP32 ONNX file by using TensorRT's ONNX parser to read the ONNX nodes and weights. It then creates an optimization profile from the fixed shape I gave it (1x3x800x800), and it then sets the build configuration to FP16 (meaning it uses FP16 kernels where possible).

- **Implement INT8 Engine**
    - An INT8 engine was built using Polygraphy with calibration on 256 representative images. Polygraph is NVIDIA's TensorRT tooling; it makes INT8 calibration and engine build reproducible and scriptable in one place. INT8 inference requires mapping real-valued tensors (activations and weights) into 8-bit integers. To do that, TensorRT needs quantization scales that describe the expected numeric ranges of activations. Calibration's job is to estimate those scales by running a set of representative inputs (satellite images) through the network and observing activation ranges. Without calibration (or without explicit quantization scales embedded into the model graph), TensorRT can't reliably decide how to compress activations into INT8. The choice of using 256 representative images was intentional, as the data loader caps calibration samples at 256, and it makes the calibration step faster and more manageable on the Jetson while still being enough to get reasonable activation ranges. Too few or unrepresentative calibration images lead to worse activation range estimates and a higher risk of INT8 accuracy loss. More images means better coverage of the data distribution, but longer calibration time, so 256 was chosen as a nice in-between trade off. The INT8 model is not a "pure" 8-bit model, though; it does have FP16 fallbacks. TensorRT will run a model layer in INT8 if that layer has a valid INT8 implementation, but if a layer cannot run in INT8, it is allowed to run in FP16 instead. The model is mixed precision since the vision transformer has operation limitations, namely Softmax (softmax normalized attention weights and produced final class probabilities). Softmax is numerically and algorithmically hard to do accurately, so trying to perform those calculations within INT8 can cause significant accuracy degradation, so Softmax is usually kept in a higher precision; in this case, FP16. The solution to obtain a pure INT8 model is to write custom INT8 softmax implementations through approximations, which requires specialized kernels, plugins, and additional engineering.
- **Write Inference Script for the INT8 Model**
    - Inference is the deployment-time forward pass of a trained model. Input images are provided, the model computes outputs, and those outputs are interpreted as detections. Inference does not update weights; it only evaluates the network to produce predictions. The inference script loads the TensorRT engine, preprocesses each image to match the model's expected input (input image size set to 800x800), and then executes the engine (the actual inference call). It then postprocesses the raw outputs into usable detections by applying softmax to the class scores and selects the best class and confidence score for each predicted box. It converts YOLOS box format to pixel coordinates, then filters by a confidence

score threshold and applies Non-Max Suppression. It then prints the detections and draws bounding boxes and labels on the input images.

- **Compute/Report Metrics for 8-bit Model**
    - Model accuracy was compared using COCO mAP@[0.50:0.95] (average precision averaged across IoU thresholds 0.50-0.95). The FP32 baseline achieved mAP=0.0958. Converting to an FP16 TensorRT engine reduced mAP to 0.0724 (about a 24.4% relative drop from FP32). The INT8 TensorRT engine achieved mAP=0.0735, essentially matching FP16. Overall, the INT8 engine remained close to FP16 while both were lower than FP32. The small difference between FP16 and INT8 is consistent with the INT8 workflow using calibration scales derived from 256 representative images, which makes INT8 quantization numerically meaningful and reduced additional accuracy loss from quantization error. In addition, the INT8 engine is also mixed precision, where some operations that cannot be performed in INT8 fallback to using FP16 which also keeps INT8 behavior close to FP16.

Discussion of Contribution to Milestone 4:

- **Sloan Hatter:** I implemented the FP16 engine model on the Jetson by compiling the FP32 ONNX model into a TensorRT engine using FP16 from NVIDIA's TensorRT framework. I then implemented the INT8 engine model using NVIDIA's Polygraphy and calibration frameworks. In order to run inference on the INT8 model, I wrote an inference script, which provided me with bounding boxes for the satellite images along with confidence scores (how confident the model is that it sorted the satellite component into the right class). Finally, I computed the metrics for the FP32, FP16, and INT8 models to gauge how well each model is performing.

Task Matrix for Milestone 5:

Task	**Sloan**
Implement 4-bit model	100%
Begin progress on 2-bit model	100%
Formal compilation of results thus far for publication (per Faculty Advisor's request)	100%

Discussion of Planned Tasks for Milestone 5:

- **Implement 4-bit Model**
    - Since the 8-bit model has been achieved, I can now move on to implementing the 4-bit model. This can be achieved by running an INT4 inference engine for the Vision Transformer model, utilizing NVIDIA's TensorRT, which provides optimized support for low-precision inference. The trtexec tool will be used to profile and convert HALO into an optimized TensorRT engine for deployment.
- **Begin progress on 2-bit Model**
    - The Jetson SDK does not have specific support for an INT2 inference engine, so most of this model will need to be implemented manually through post-processing scripts, most likely using Post-Training Quantization (PTQ).

- **Formal Compilation of Results thus far for Publication**
    - Per my faculty advisor's request, I will begin a formal compilation of each HALO model metric, from the 32-bit to the 4-bit model.

Date of Meetings:

- 02/12/26
- 02/19/26

Client Feedback on Milestone 4:

See Faculty Advisor Feedback below.

Faculty Advisor Feedback on Milestone 4:

Faculty Advisor Signature: _____ Date: 19 Feb 2026