Real-time CNN-based Spacecraft Pose Estimation on COTS Reconfigurable MPSoC

IEEE NEWCAS 2024
Paper ID 9081

Julien Posso*, Guy Bois*, Yvon Savaria†
Department of Computer Engineering* / Electrical Engineering†, École Polytechnique de Montréal, Quebec, Canada

INTRODUCTION

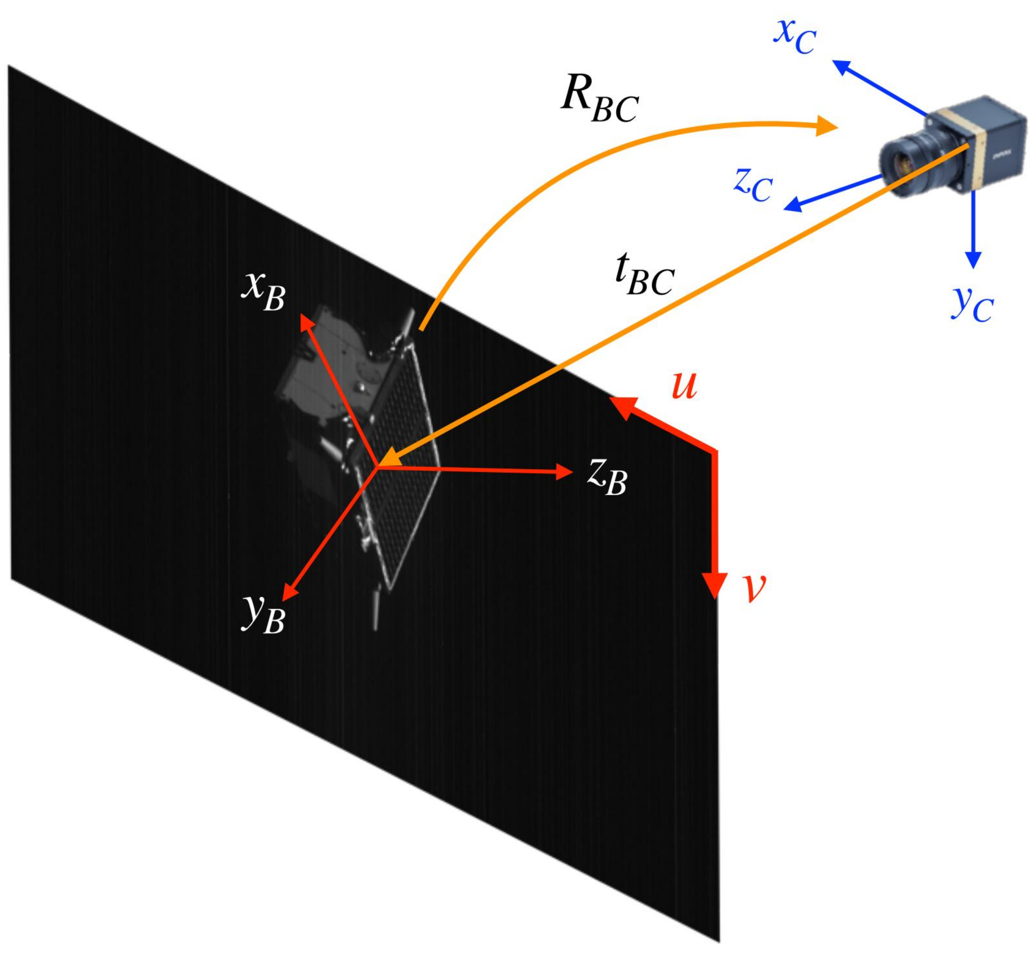


Figure 1: Illustration of the spacecraft pose estimation task [1]

Problem: Estimating the pose (i.e., the relative position and orientation in space) of a known, uncooperative spacecraft from a monocular image.

Motivation: Enhancing autonomy in space missions, such as formation flying, autonomous docking, and debris removal.

Existing Literature: Many neural network-based methods exist for solving spacecraft pose estimation, but challenges persist in achieving real-time inference.

CONTRIBUTIONS

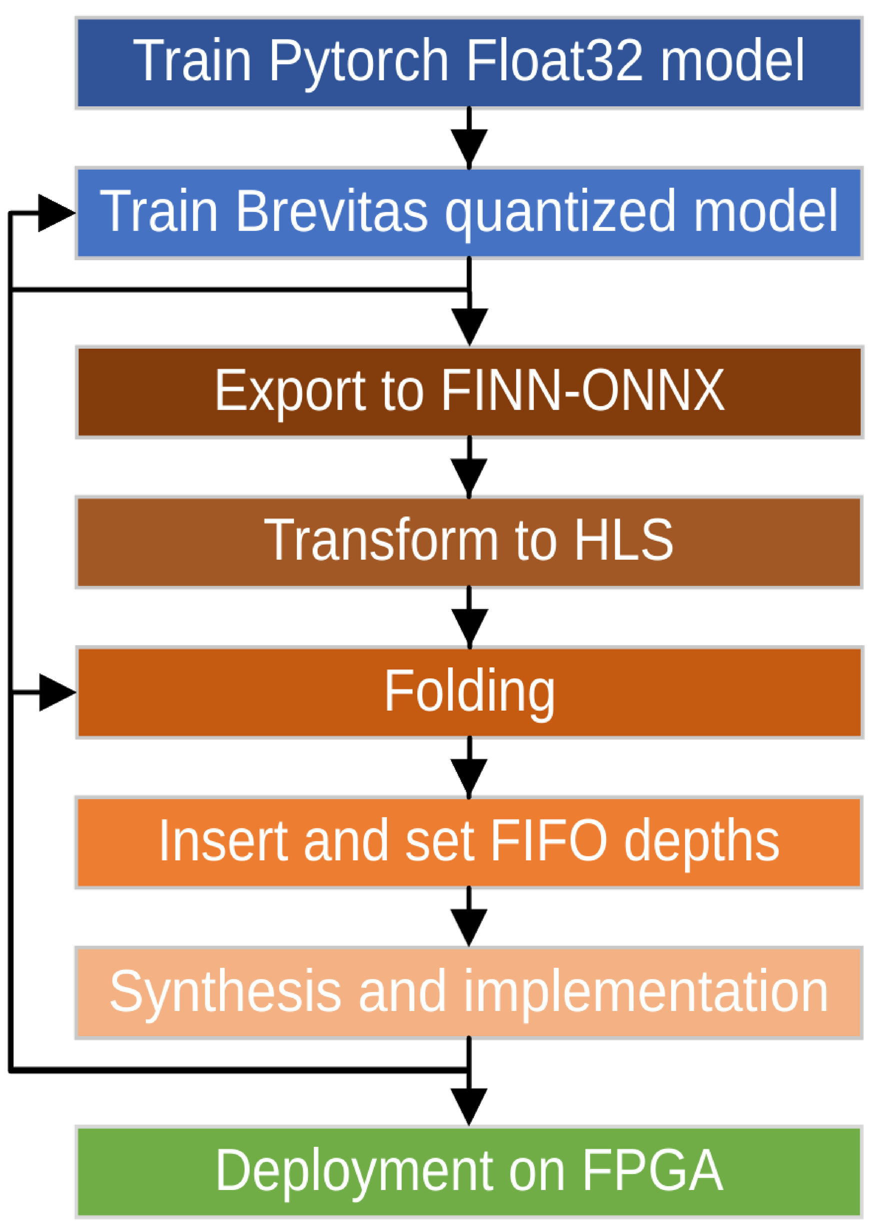
First Real-Time Implementation: The first real-time implementation of a popular spacecraft pose estimation neural network on a space-proven Xilinx UltraScale+ MPSoC (ZCU104 evaluation board).

Open-Source Contribution: The first open-source, real-time spacecraft pose estimation implementation based on neural networks (scan QR code below).

Performance: Our FPGA dataflow accelerator is 7.7 times faster and 19.5 times more energy-efficient than the best-reported values in existing spacecraft pose estimation literature.

Quantization: An in-depth study of mixed-precision quantization and methodology for bit-width selection.

GLOBAL METHODOLOGY



Pre-train a Float32 model to save time during Quantization Aware Training (QAT).

QAT: Layer-Wise Bit-Width Selection. See the next section for more details.

Export the neural network to a graph. Each neural network layer is represented as a node.

Apply transformations to the graph so that it only includes HLS-compatible nodes.

Set the parallelism level of each node to control the throughput and FPGA resource usage.

Insert FIFOs between nodes of the graph and set their size.

If FPGA resources are exceeded, reduce parallelism (folding) or bit-width (QAT).

The result is a pipeline of units connected by FIFOs, each accelerating a neural network layer.

Figure 2: The proposed methodology: from neural network training in PyTorch/Brevitas to FPGA on-board implementation with FINN

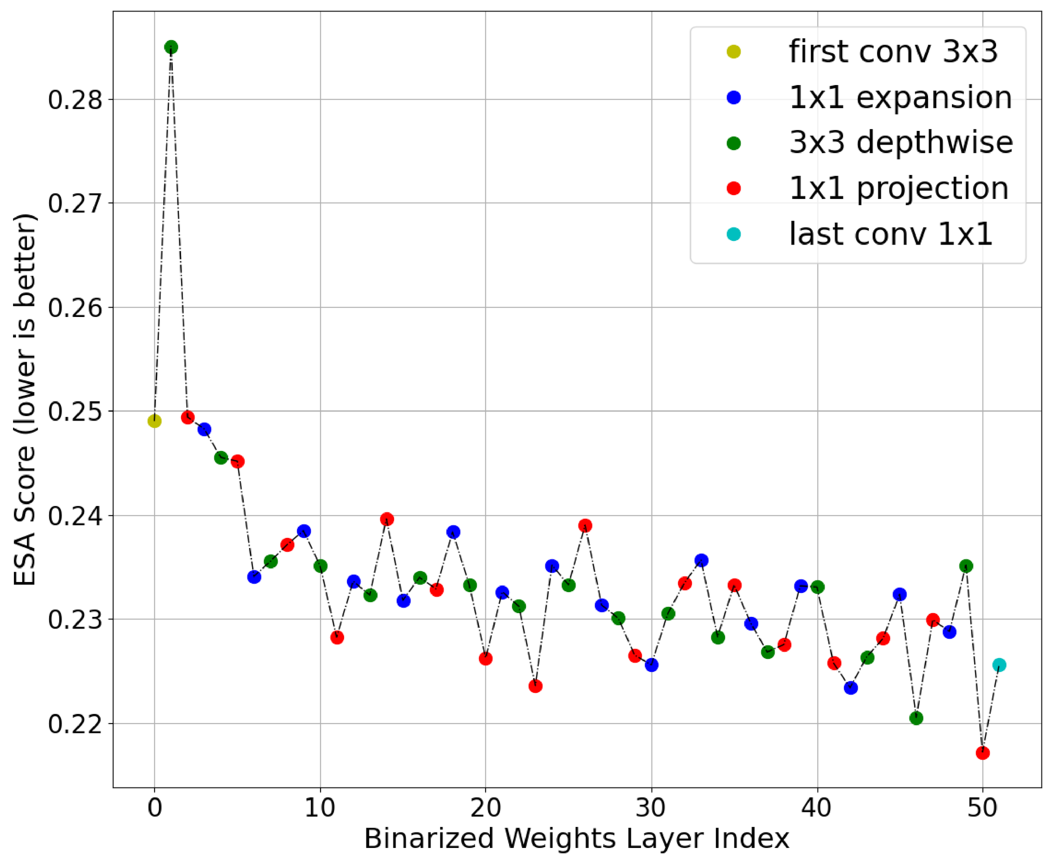
Initial Model: Trained Float32 Mobile-URSONet model with a MobileNetV2 backbone.

Quantization: Quantization-Aware Training using per-channel symmetric uniform quantization with layer-wise arbitrary precision.

FPGA Implementation: Custom dataflow accelerator using the FINN library. On-chip weights and activations minimize latency and energy consumption. Operates at 187.5 MHz. Utilizes 91% LUTs, 90% BRAMs, 32% DSPs, and 26% Flip-Flops of the FPGA.

Limiting Factors: LUTs for computing convolutions and BRAMs for storing activation functions.

MIXED-PRECISION QUANTIZATION



Proposed Methodology to evaluate the sensitivity of a neural network layer's weights to quantization

- Train the neural network 51 times
- In each training iteration, binarize the weights of one specific layer while keeping all other layer weights at 8 bits.
- Different colored markers represent different types of layers, providing a clear comparison of their quantization sensitivity.

Figure 3: ESA score (lower is better) when only one convolutional layer's weights are binarized, with all other layers and activations at 8 bits

Sensitivity to Quantization: Depends on the number of parameters of the layer.

Chosen Bit-Width Configuration: Balancing throughput, accuracy, and FPGA resource usage.

- **4-bit Activation Functions**: FPGA resource usage increases exponentially with bit-width. Above 4 bits, resource usage becomes excessive; below 4 bits, accuracy degrades.
- **Mixed-Precision Weights**:
  - 1st convolution (3x3): 4 bits
  - 2nd convolution (3x3 depthwise): 6 bits
  - 3rd convolution (3x3 projection): 4 bits
  - All other layers: 3 bits

The most sensitive layers require greater precision

RESULTS AND COMPARISON

Table 1: Measured power consumption, throughput (FPS) and energy efficiency compared to the existing literature

Implementation	Power (W)	FPS	FPS per Watt
FPGA ZCU104 (ours)	0.865	58.7	67.9
CPU Intel Atom [2]	3.7	6.58	1.78
FPGA Ultra96 [3]	1.32	x	x
FPGA Ultra96 [4]	x	167	x
Google edge TPU [5]	2.2	7.66	3.48

Key result: our accelerator is 7.7 times faster and 19.5 times more energy efficient than the best previously reported spacecraft pose estimation implementation.

REFERENCES

[1] M. Kisantal, S. Sharma, T. H. Park, D. Izzo, M. Märtens and S. D'Amico, "Satellite Pose Estimation Challenge: Dataset, Competition Design, and Results," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 5, pp. 4083-4098, Oct. 2020
[2] K. Black, S. Shankar, D. Fonseka, J. Deutsch, A. Dhir, and M. R. Akella, "Real-Time, Flight-Ready, Non-Cooperative Spacecraft Pose Estimation Using Monocular Imagery," arXiv:2101.09553 [cs], Jan. 2021, arXiv: 2101.09553.
[3] K. Cosmas and A. Kenichi, "Utilization of FPGA for Onboard Inference of Landmark Localization in CNN-Based Spacecraft Pose Estimation," Aerospace, vol. 7, no. 11, p. 159, Nov. 2020, number: 11 Publisher: Multidisciplinary Digital Publishing Institute.
[4] S. Wang, S. Wang, B. Jiao, D. Yang, L. Su, P. Zhai, C. Chen, and L. Zhang, "CA-SpaceNet: Counterfactual Analysis for 6D Pose Estimation in Space," Jul. 2022, arXiv:2207.07869
[5] A. Lotti, D. Modenini, P. Tortora, M. Saponara, and M. A. Perino, "Deep Learning for Real Time Satellite Pose Estimation on Low Power Edge TPU," Jun. 2022, arXiv:2204.03296