Analysis and Implementation of Space Avionics Co-Processor for Deep Learning Acceleration

European Data Handling & Data Processing Conference for Space
dgarjona@gmv.com

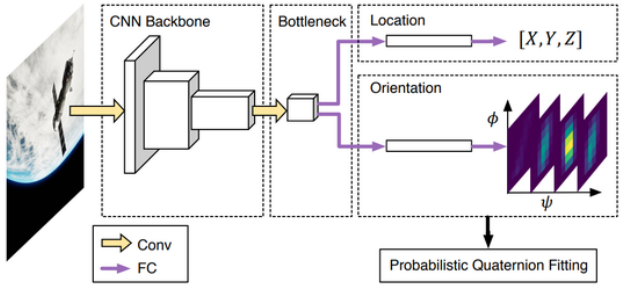
D. Gonzalez-Arjona
J. Ferre
A. Jiménez-Peralo
D. Fortun
D. Lo Presti
D. Gogu

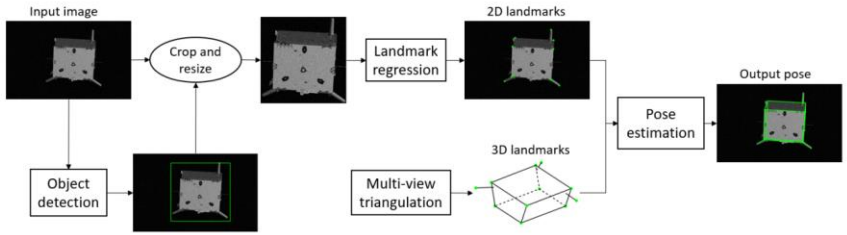
EDHPC 2023
European Data Handling &
Data Processing Conference for Space
2 - 6 October 2023 | Juan-Les-Pins | France

Space-Based Deep Learning

On-Board Processing

- In recent years, Deep Learning (DL) is gaining popularity in the space sector as a versatile tool for EO classification and detection problems and Vision-Based GNC systems, among others.
- Autonomous vision-based spacecraft navigation is one key area with the potential of largely benefiting from Deep Learning estimation methods
- More precisely, problems involving detection, segmentation or classification can have a significant improvement in performances and robustness by switching to an AI-based approach.
- Edge-Computing of inferred DNN, CNN or a whole continual Learning application imposes high-performance processing requirements for autonomous real-time execution in space avionics





Avionics for DL and Deployment

High-Performance Acceleration

a) FPGA HDL handwritten ad-hoc IP
   a) General DL IP

b) SW-based
   a) CPU
   b) FPGA HLS Flow

c) Dedicated AI cores
   a) VPU
   b) TPU
   c) AI engines

d) Model-Based Design options
   a) FPGA, CPU, DPU, DSP, GPU

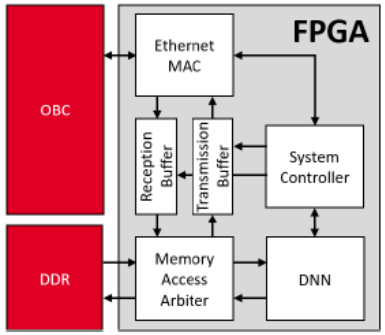


Figure 4: Avionics diagram
architecture of hand-written HDL-coded GMV's Deep Learning FPGA accelerator

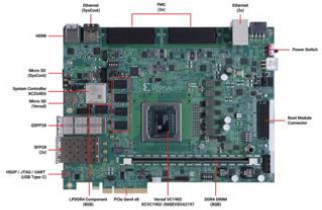
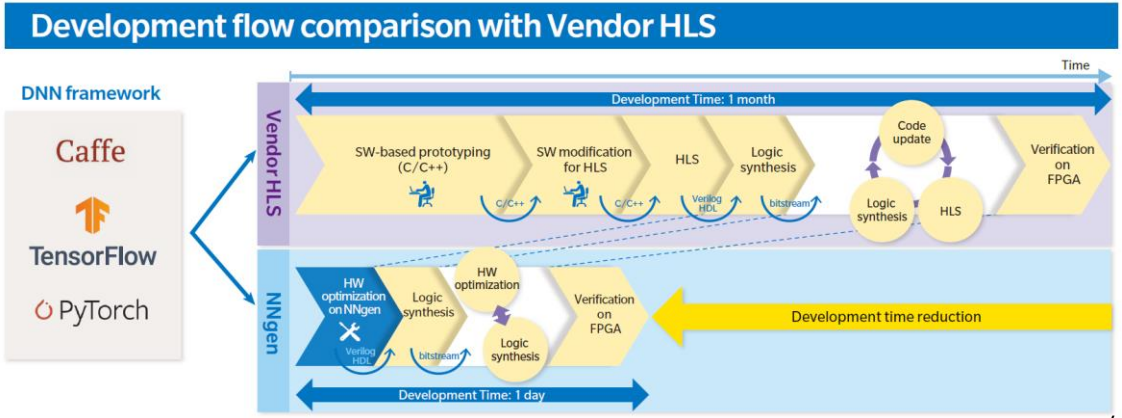


Fig. 7. Board Features of the Versal AI Core VCK190 Evaluation kit

ESA test of Intel Myriad 2 AI chip at SPS North Area



Source: https://research.konicaminolta.com/en/technology/tech_details/nngen/

DL Accelerator Development and Analysis

Processor, tools, deployment, architetcure

- Flight-segment AI accelerator compatible with new standard ADHA architecture
  - High-Performance dedicated Processing Unit
  - COTS, Ruggedized, Rad-Tol, Rad-Hard

- Different approaches being developed
  - (A) FPGA hand-written RTL DL processor
  - (B) Model-based Design to HW/SW FPGA SoC
  - (C) Dedicated AI cores:
    - Ubotica's Myriad2 VPU + Controller FPGA
    - [bonus-track] Versal ACAP AI core

- AI Demonstrators Use-Cases
  - Moon Landing Crater Detection
  - Space Exploration Asteroid patch pinpointing
  - Space-Based Surveillance Debris Detection

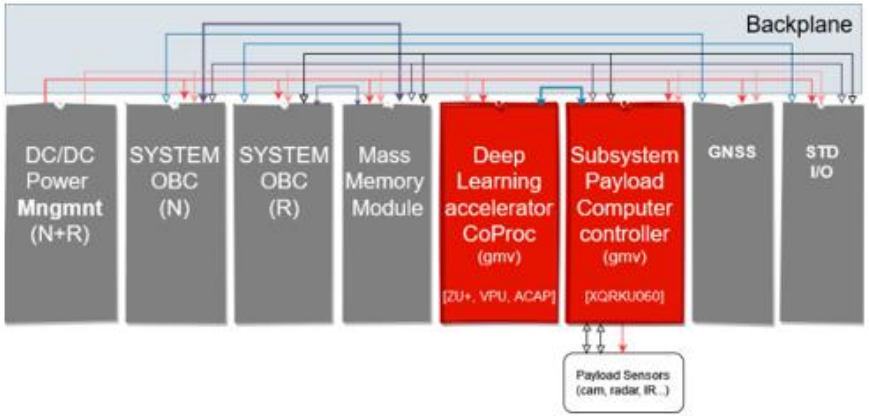
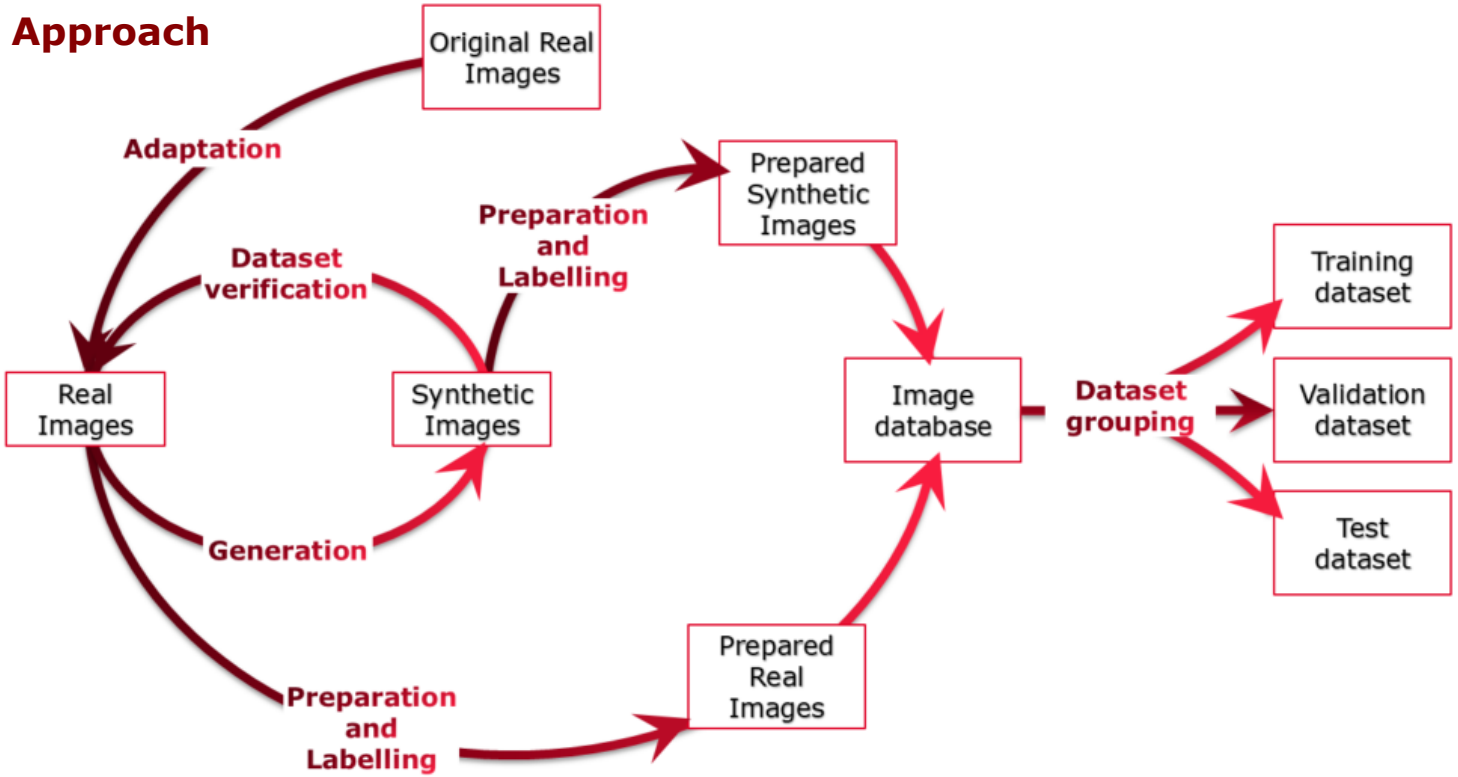


Fig. 1. Deep Learning coprocesor accelearator into ADHA configuration

Generation of image datasets

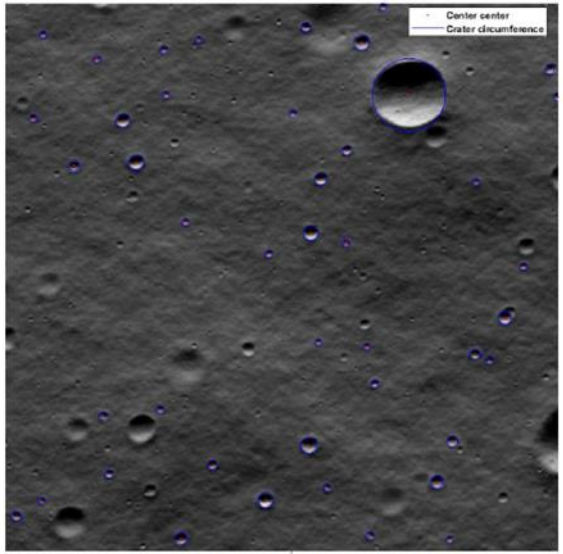
General Approach



FPGA DL processor RTL (A)

Scenario and Objective

- Use-Case: Crater Edge Detection for Moon landing Absolute Nav
- DL Model provided as input using TensorFlow and Keras
- Many challenges to port a full accurate Float64_t, images 2048x2048
- Python library created to extract layers, dimensions, weights, biases
- Avionics
  - ALPHA-DATA-KU060 representative of rad-tol XQRKU060
  - Dedicated RTL architecture from scratch



AI based Absolute Navigation HW results, top left corner zoomed area

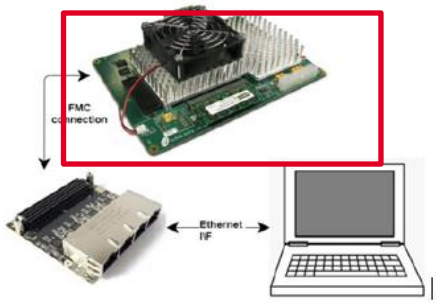
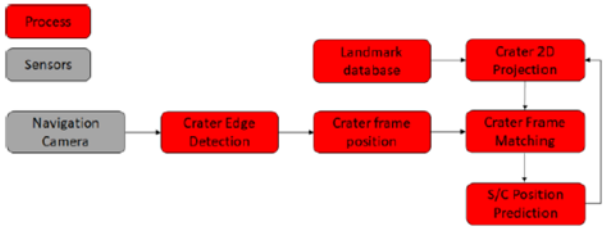


Fig. 5. FPGA-based deep learning set-up of the demonstrator

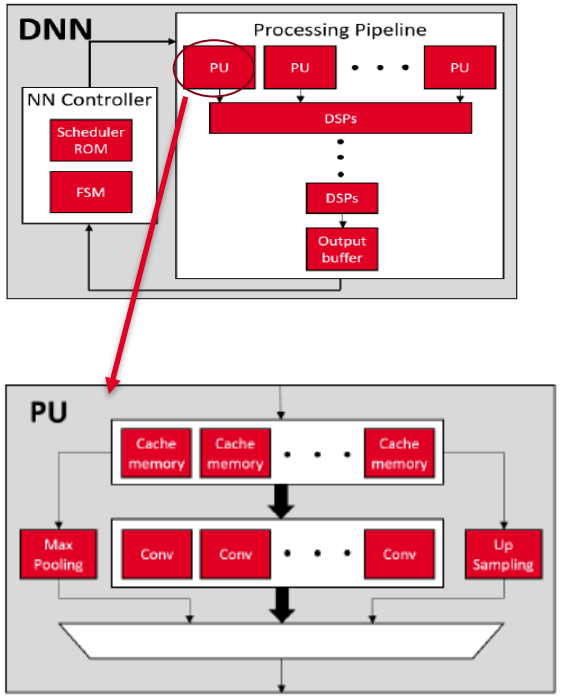


AI-based Absolute Navigation architecture

Technical Details (A)

FPGA dedicated implementation

- Ad-hoc architecture for on-board deep neural networks in HDL
- Reconfigurable FPGA-accelerated deep learning processor.
- parameterization of modules programmatically generated and generic models be constructed from implemented layer types
- Controller HDL module sequences steps and handles data transfers
- Processing pipeline implements different processing units interconnected
- Processing Units (PU) implements functions, convolution kernels and other operations over data
  - Diferent generics used in HDL to adapt to the Neuronal Network Architecture, functions and sizes (pre-synthesis of RTL)
  - DSPs in an inverted pyramid for bias and mapping of activation functions ReLu/sigmoid



Demonstration Results (A)

FPGA dedicated implementation

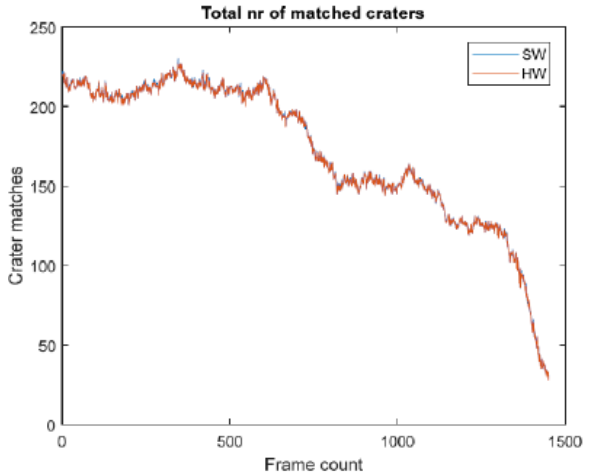
- Data: 2048x2048 8-bit pixels input images and 2048x2048 pixels output feature map
- U-Net model containing 30 layes with Conv2D, DepthwiseConv2D, MaxPool2D, concatenation in 14 PUs and with different activations ReLu and sigmoid.
- BRAMs optimization based on simplification to low-down from 96% to 72% Resources utilization
- The estimated center and radius of the craters error with respect to SW results were found to be subpixel.
- 98% of crater matches of SW implementation vs FPGA one.
- Demonstrator over ethernet connection to Monitoring PC imposed many communication problems and reduced speed link

TABLE I. U-NET MODEL RESOURCES UTILIZATION IN XQRKU060

Resource type	Available	Total	Percentage
FFs	663360	98419	15%
LUTs	331680	115739	35%
BRAMs (36Kb)	1080	780	72%
DSPs	2760	1136	41%

Center of radius comparison of FPGA vs SW solution

	Center detection precision on x axis	Center detection precision on y axis	Radius detection precision
Mean	0.83	0.75	0.29

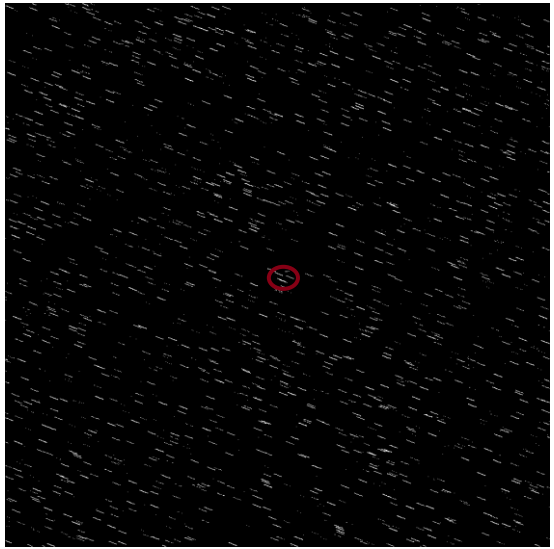


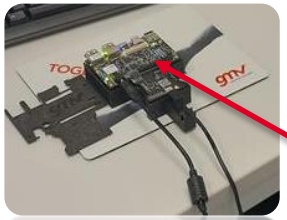
Model-based Design on SoC (B)

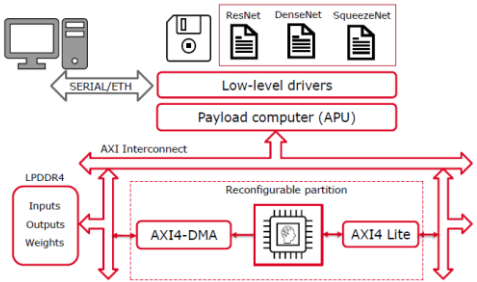
Scenario and Objective

- Use-Case: space debris detection and localization in space-based surveillance system using AI
- Different Model-Based Design tools evaluated: from Model to HW
  - Intermediate representation languages to prepare model towards HLS input coders
- Each MBD toolset includes a different HDL autocoder:
  - Xilinx HLS, BAMBU and Veriloggen
- Avionics
  - Ultra96 Zynq Ultrascale+ ZCU03 representative of commercial fligth cubesat boards
  - FPGA Dynamic Partial reconfiguration framework to switch between implementations

debris detection



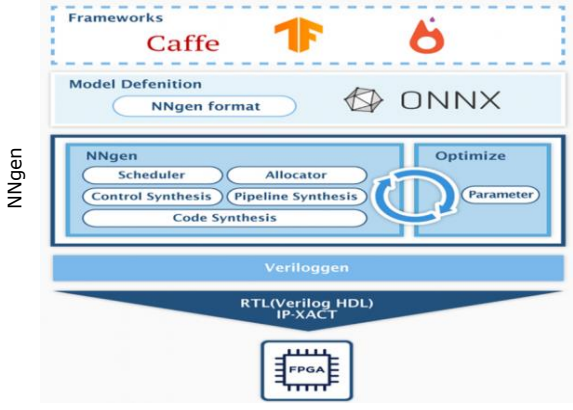


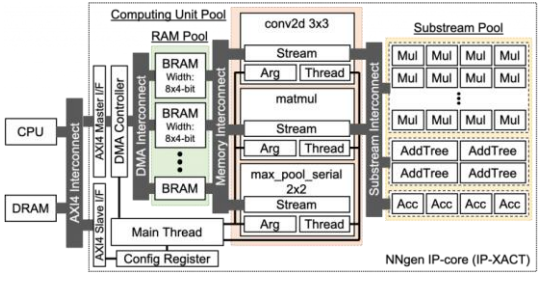


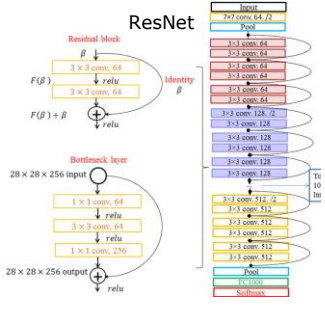
Technical Details (B)

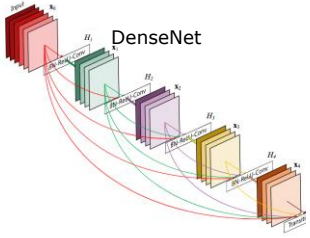
MBD from concept to SoC HW

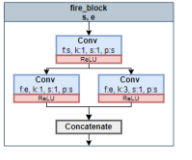
- Comparison of hls4ml, SODA-OPT, NNgen
- Solution deployment using NNgen open-source, python-based tools
- Initial model from Caffe, TensorFlow, Pytorch or direct NNgen format primitives
  - ONNX intermediate representation if not using primitives
  - We used Pytorch→ONNX→Veriloggen→RTL.
- Four models: ResNet, DenseNet, SqueezeNet and custom TinyCNN
  - TinyCNN, is a shallower architecture made ad-hoc, that was shown to perform relatively well for the space debris detection task
- Data-types selection and quantization for inputs (mean, std), bias, weights
- Verilog HDL source and IP-XACT IP-core package (AXI4-lite & AXI4 full)
- Veriloggen Backend open-source
  - (modifiable to port to Xilinx, Microsemi, NanoXplore, FG-Lattice…)
- 10x speedup Performance vs SW:
  - VGG-11 & ResNet on ImageNet, Digit classification

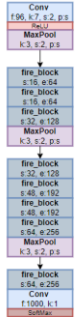












Demonstration Results (B)

MBD from concept to SoC HW

- about 2/3 of the total FPGA area resources, TinyCNN less resources, comparable to ResNet but for the extense BRAMs
  - Dynamic partial reconfiguration is used to switch between different accelerators in runtime.
- Accuracy functional results above 97% but for TinyCNN ~84%
- Deployment in Zynq Ultrascale+ provides different deployment processing options: PL FPGA, ARM A53 quad-core, ARM R5 dual-core
- For the ARM SW processors, both use active data and instructions caches, float-point and the A53 takes advantage of ARM Neon SIMD extensios
- Execution time:
  - fastest for TinyCNN in all the resources (FPGA or processors) and even performs better in SW quad-core than in FPGA acceleration
  - ResNet penalized in SW procesors compared to the FPGA to SW comparison in other models (speed-up above 18x)

* the measurements of time for the FPGA hardware accelerators include the overheads due to the memory accesses to the input images and the network weights by the accelerator

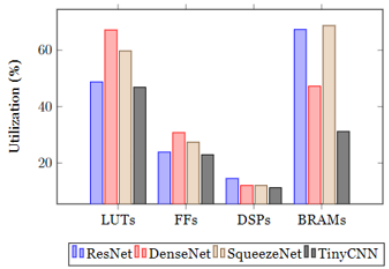


Fig. 2. Resouce utilization of AI model accelerators on Zynq US+

TABLE II. ACCURACY FOR EACH AI ACCELERATOR, FLOATING-POINT PRECISION OBTAINED IN PYTORCH, AND INTEGER PRECISION IN HW

Parameter	ResNet	DenseNet	SqueezeNet	TinyCNN
Floating point avg accuracy (%)	99.60	98.20	99.51	85.14
Integer avg acuracy (%)	98.0	97.0	98.0	83.0
Floating-point avg IOU	0.88	0.73	0.82	0.77
Integer avg IOU	0.61	0.64	0.54	0.73

TABLE III. EXECUTION TIME IN MS AND SPEED-UP RESULTS FOR EACH AI ACCELERATOR IN DIFFERENT HW TARGET DEPLOYMENT OF THE ZYNQ ULTRASCALE+ RESOURCES

Parameter	ResNet	DenseNet	SqueezeNet	TinyCNN
PL HW WCET	1456 ms	706 ms	1171 ms	41 ms
A53 WCET	26484 ms	3578 ms	4568 ms	21 ms
R5 WCET	30209 ms	22897 ms	17215 ms	791 ms
PL HW avg inference ET	1456 ms	706 ms	1171 ms	41 ms
A53 avg Inference ET	26444 ms	3566 ms	4560 ms	21 ms
R5 avg Inference ET	30208 ms	22896 ms	17211 ms	791 ms
A53 Speed Up	18.18	5.06	3.89	-
R5 Speed Up	20.72	32.42	14.69	19.41

Dedicated AI cores (C)

Scenario and Objective

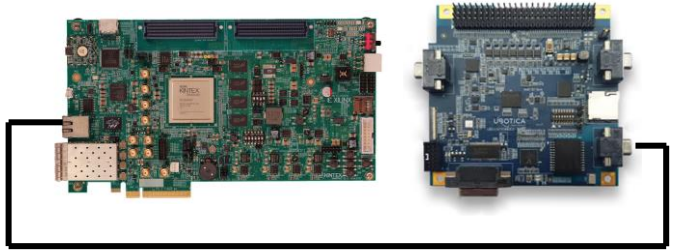
- Use-Case #1: autonomously detect and provide list of craters and positions with accuracy performances for dencent and landing on Moon(7,3km-100km)

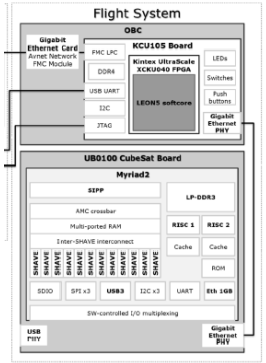
- Use-Case #2: autonomously detect and pinpoint previously selected patches on images of the surface of the asteroid for scientific exploration

- Avionics

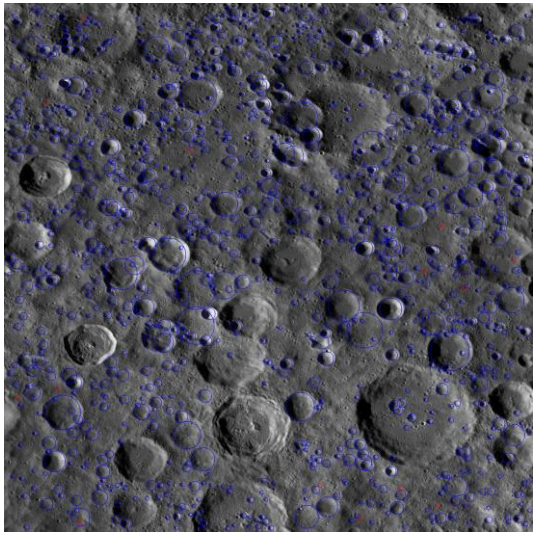
  - KCU105 Kintex Ultrascale representative XQRKU060 with LEON5 softcore and dedicated FPGA IPs

  + UB0100 CubeSat Board + Movidius Myriad2 VPU

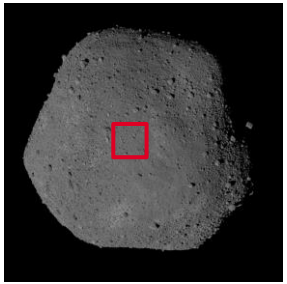




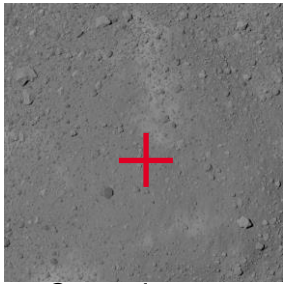
crater detection



Patch pinpointing



Reference image



Query image

Technical Details (C)

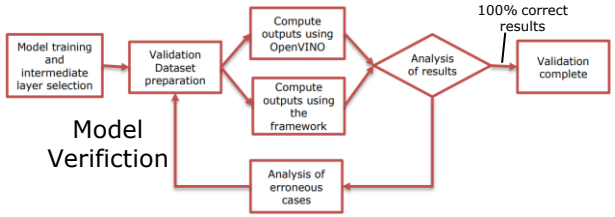
VPU dedicated AI engines

- Myriad2 VPU for use in low-power edge applications providing in excess of 1 TOPs of compute power, integrated in UB0100 Ubotica Cubesat board characterized for use in space.
  - Intel Movidius Myriad2 USB stick for development steps

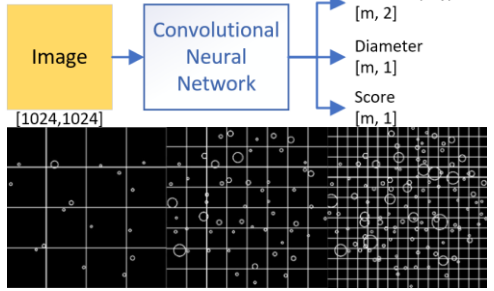
- Reconfigurable FPGA Kintex Ultrascale implements LEON5 acting as Subsystem Payload Controller interfacing to OBC, camera sensor and linking with UB0100 through non-space ethernet interface.
  - Implements pre-processing camera images step, configuring and monitoring VPU accelerator

- GMV developed DNN design models for both Use-Cases #1 and #2
  - Use-case#1 YOLOv3 modified architecture with grid of square cells division of images in different scales
  - For Use-Case#2 it relies on Siamese network to compare and measure similarity of reference patch and query image from camera. Trained with triplet los function

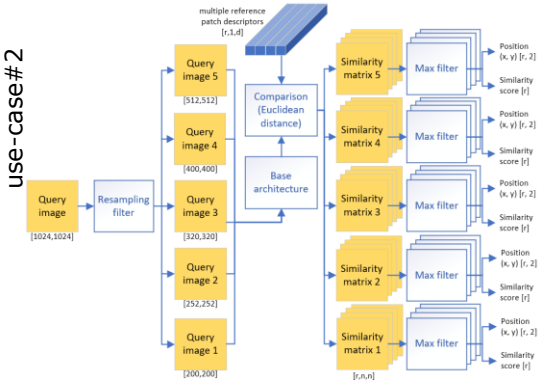
- OpenVino intermediate framework from model to dedicated Myriad2 library. Ubotica developed the low-level Myriad2 SW and its optimization



use-case#1



use-case#2



Demonstration Results (C)

VPU dedicated AI engines

- Demonstration of 2 use-cases. Receiver operator curve (ROC) adapted to localization problem (normally classification problem) to use in 2 UCs
- UC#1 NN Moon D&L stochastic error of 7.6 pixel in X and Y synthetic data and 7.67 real dataset (True Positive rate > 0,3)
- UC#2 NN Asteroid patch stochastic error of 43 pixel validation dataset while 74 in test dataset with different terrains (True Positive rate > 0,3)
- Execution time: 13.56 seconds are used to complete chain reading the image, built the collage, perform the inference on it, obtain the descriptors ordered correctly, same process with the patch and calculate the Euclidean distance.

ROC NN Moon Descent and Landing Use-Case #1

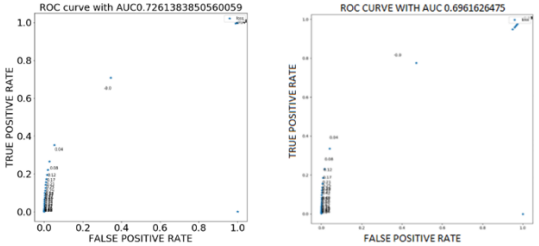


Fig. 4. Roc curve of synthetic images (left) and aggregated ROC curve of all LRO images(right).

ROC NN Asteroid patch pinpointing Use-Case #2

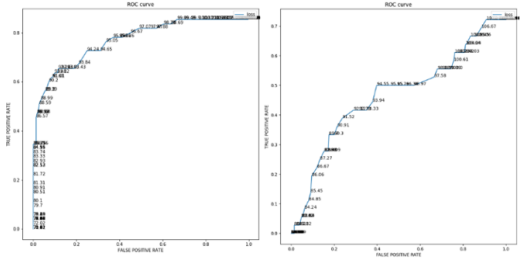


Fig. 6. Roc curved computed for the validation test of sythethic dataset (left) and real dataset(right)

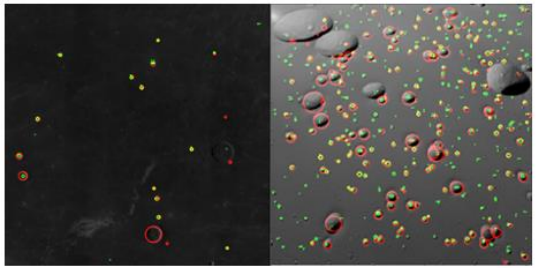


Fig. 3. Crater detection performed by AI executed on Myriad2 VPU; (left) results over LRO image; (right) results over PANGU image

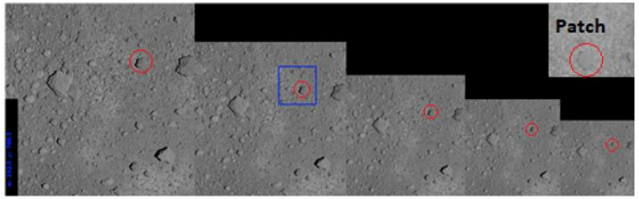
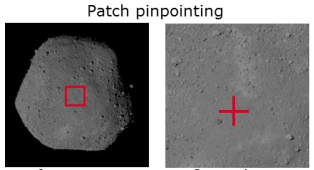


Fig. 5. Patch recognition over a mosaic of the same synthetic query image at different scales representing different ranges to the asteroid; (red circle) rock recognizable by human eye; (blue square) query image area in which CNN recognized the patch

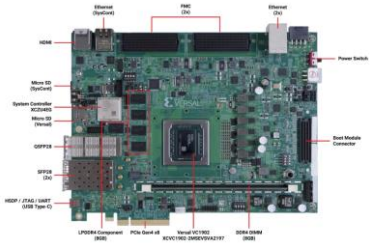
Dedicated AI cores (C-bonus)

Scenario, Avionics and Development Steps

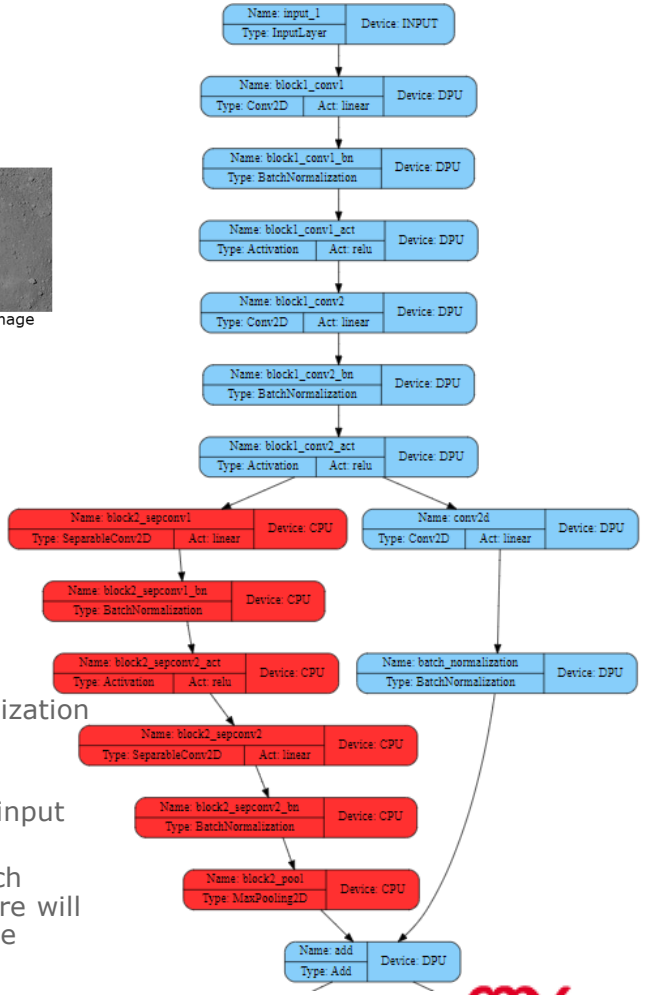
- Use-Case #2: detect and pinpoint selected patches on images of asteroid for scientific exploration



- Avionics
  - Versal AI-core VCK190 development kit (XQRVC190 rad-tol equivalent)
    - scalar processing (CPU – ARM Cortex)
    - Vector processing (DSP)
    - Programmable logic (FPGA)
    - Deep learning processing unit (DPU)
      - AI engines (scalar RISC processors)



- Development steps:
  - .h5 representation of the model generated and trained in Tensorflow2
  - Float32, sigmoid and relu, Conv2D, SeparableConv2D, MaxPooling2D, BatchNormalization and Add layers
  - Vitis AI library for inspection, followed by quantization and networks checks
  - Inspector requires target (Zynq, Zynq Ultrascale+ or ACAP), model and size of NN input
  - the .svg image represents in blue and red which layers can be quantized and which cannot. This files show that whenever the network splits in two branches, the branch guided by a SeparableConv2D layer will be totally executed by the CPU and therefore will not be quantized/accelerated, because the DPU on the VCK190 does not support the separable convolution layer



CONCLUSION

Demonstration of different deep learning avionics solutions for space applications

FPGA Dedicated Implementation (A): We introduced a specialized architecture for on-board DNN inference implemented on radiation-tolerant FPGAs. Through Hardware Description Language (HDL) development, we created a reconfigurable FPGA-accelerated deep learning processor. The architecture encompassed controller and processing pipeline elements, demonstrating its potential through the implementation of a U-Net model for crater detection. The design was resource-efficient, and laid the groundwork for further optimizations

Model-Based Design Workflow (B): We explored two use cases that leveraged model-based design workflows for accelerating DNNs on FPGAs. We evaluated different tools, such as hls4ml, NNgen, and SODA-OPT, for automatic code generation and deployment. The workflow facilitated seamless translation from algorithmic frameworks to FPGA implementations. By testing various models and configurations, we demonstrated the feasibility and potential advantages of this approach.

Video-Processing Unit and AI Engines (C): We delved into the application of dedicated accelerators, such as the Myriad2 VPU and the newly introduced Versal AI Core ACAP, for space-related tasks. Our focus was on scenarios involving crater localization on the Moon and patch detection on asteroid images. We demonstrated the capabilities of the Myriad2 VPU in detecting craters and specific patches. Additionally, we outlined the potential of the Versal AI Core ACAP, highlighting its heterogeneous architecture and AI Engine capabilities.

Thank you

INNOVATING SOLUTIONS