MACHINE LEARNING APPLICATION BENCHMARK FOR IN-ORBIT ON-BOARD DATA PROCESSING

Max Ghiglione[1,*], **Amir Raoofy**[2], **Gabriel Dax**[2], **Gianluca Furano**[3], **Richard Wiest**[1], **Carsten Trinitis**[2], **Martin Werner**[2], **Martin Schulz**[2], **Martin Langer**[4]

[1]*Airbus Defence & Space GmbH*
[2]*Technical University of Munich*
[3]*European Space Agency*
[4]*Orbital Oracle Technologies GmbH*

ABSTRACT

This paper introduces the MLAB project, a research and development activity funded by ESA General Support Technology Programme under the lead of Airbus Defence and Space GmbH with the goal of developing a machine learning application benchmark for space applications. First, the project is introduced, and examples of applications are described including their design challenges. The benchmark specifications is described, and different processing platforms to be submitted are presented. Finally, initial performance results for first submissions are shown as examples.

1. INTRODUCTION

Machine **L**earning (ML) is increasingly used in space demonstration missions, like e.g. ESA's ϕ-sat [20], paving the way towards its mainstream use in Smallsats and in large institutional projects. This trend is fueled by the use of **C**ommercial-**O**ff-**T**he-**S**helf (COTS) solutions and the improvement of tools for ML deployment on radiation-tolerant processing units. The satellite industry has been looking at these developments with great interest.

For successful deployment and use, however, we are facing three main challenges: 1) on-board spacecraft hardware has limited processing capabilities, requiring algorithms to be optimized for a specific embedded hardware platform; 2) the integration into the industry workflow requires new sets of tools and interactions between developers; and 3) the available datasets in terms of open accessibility and reusability for space missions are limited, as data is either proprietary or poorly labeled. To address these challenges, and to enable the needed in-depth evaluation of these techniques for use in on-board applications, we need a comprehensive benchmarking approach specifically targeting the challenges connected to ML inference applications in space.

The commercial word has already pioneered such benchmarks, in particular with MLPerf [18] and its embedded counterpart, proving the importance of such a method for ML inference.

In the frame of the **MLAB**[1] project, Airbus Defence & Space, Technical University of Munich, and OroraTech are working on developing a novel ML inference benchmarking approach based on the commercial solutions and MLPerf method [18].

This approach provides the ability to simplify the comparison of algorithms in the early phases of their development, enabling engineers to predict the necessary processing power for the intended applications (1). Further, combined with appropriate benchmarking frameworks and codes, it will enable the investigation of various software tools, diverse and custom reconfigurable IP designs, and COTS solutions for ML inference targeting on-board data processing (2). Finally, we will rely on public datasets or publish the needed datasets, supporting the wider research in the field and enabling full reproducibility (3).

The benchmark suite will cover a diverse set of algorithms, e.g., feature extraction, object detection, classification, tracking, and change detection of different complexity. This ensures that the full breadth of space use-cases and different computational complexities are represented in benchmarks. Moreover, the benchmark is intended to cover a diverse set of ML models including various classical convolutional models with different network architectures in terms of complexity and size. This diversity ensures the coverage of various models with different computational complexity and memory requirements. The benchmarking suite mainly relies on publicly available large-scale standardized datasets to ensure the reproducibility of results. Specifically, the datasets that have been published in recent years, including BigEarth, Airbus Ship Dataset, Cloudnet, and EuroSAT, are promising candidates. In addition, the benchmark approach covers various ML inference development and deployment tools. Due to the fact that optimization plays an important part in the final inference performance, tool choice

*Contact: max.ghiglione@airbus.com

[1]**M**achine **L**earning **A**pplication **B**enchmark

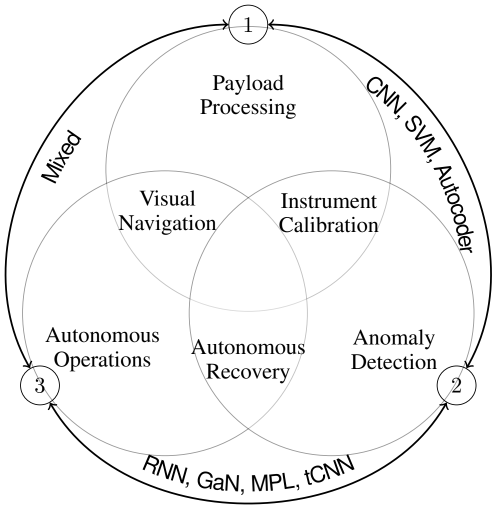

Figure 1: Machine Learning Application Examples

is crucial to meet the performance requirements with a minimum amount of design effort. For this reason, we will include inference tools, like FINN [5], Vitis AI [6], hls4ml [7], VectorBlox [8], and more, featuring different levels of HW design. Additionally, we cover the *general* performance metrics associated with the payload applications that use ML inference, including accuracy rates, throughput, model complexity, computational complexity, resource utilization, and memory footprint. Finally, we also address the *space-specific* objectives of on-board computation that demand more careful consideration and implementation, such as energy efficiency or tail-latency bounds.

2. ML INFERENCE IN SPACE

Figure 1 shows a high-level view of artificial intelligence (AI) in on-board processing applications, also called *edge* AI. We identified three main categories of applications: payload processing, anomaly detection and the wide field of autonomous operations, including visual navigation and autonomous decision making of spacecraft.

On one side, non-critical payload applications (2) have a compelling business case promising the reduction of downlink data through a wide range of on-board processing tasks, starting from simple image classification to compression use cases or multi-class segmentation of sensor data. While many applications can benefit from simpler ML techniques (like SVM[2]), a great interest is currently in the inference and benchmarking of neural networks, driven by the raise of standard models like ResNet[3] and transfer leaning.

On the other side, the improvement of anomaly detection (1) and prediction on spacecraft draws significant attention. Such tasks lie in the field of applications critical to the operation of satellites and will likely employ radiation-hardened CPU- or FPGA-based processing. Based on literature, most applications rely on time-series analysis using RNN[4] or GAN[5] to improve the training.

Autonomous operations (3) are another set of applications and are a field of interest due to the complexity of upcoming constellations and costs of ground operators. Automating and increasing the speed of decision making on satellites will relax requirements on other on-board components and subsystems while possibly improving operational capabilities. Moreover, AI will be the enabler for completely new missions. For example, when segmentation and detection of objects of interest could be proven on space-viable hardware and be used for autonomous operations, a satellite could perform focus sensing autonomously on targets to either improve data quality or latency of the information to the end user. Another example is the capability of real-time detection of Oil spills, where alerts can be sent before the sensing data is downlinked and reviewed by the ground operators. Such applications are certainly more complex than standalone anomaly detection or cloud classification algorithms and naturally more complicated than the benchmark. However, proving the maturity of the underlying design techniques through benchmarking will be a step towards fully AI-driven missions.

3. CHALLENGES OF MACHINE LEARNING IN SPACE

The adoption of new technology on-board satellites is still strongly limited by the requirements of reliability and availability, which traditionally have imposed the use of components with flight-heritage and extensive qualification. This is why on-board processing on space-borne data systems still relies on old components that do not provide enough computational power to run most of the innovative state-of-the-art algorithms [21]. The goal of MLAB activity is to narrow down a set of performance metrics that can be used to compare the different algorithms and processing platforms and help with the selection of suitable COTS components for rapid spin-in for space missions.

Limited on-board resources are one of the issues common to all electronic equipment in space. Neural networks are especially challenging in terms of memory imprint due to the number of weights used in the model. Even modern **O**n-**B**oard **C**omputers (OBCs) have cache memories that are too small to store all the weights of the network, potentially causing bottlenecks in terms of data transfers to off-chip memory which impacts a theoretical calculation of the inference time. On an FPGA or ASIC, memory access can be tailored in necessary with a dedicated memory. Moreover, contrary to neural networks for image processing, anomaly detection networks could potentially be stored internally on high-end FPGAs using

[2]Support-Vector Machine
[3]Residual Networks
[4]Recurrent Neural Networks
[5]Generative Adversarial Networks

on chip memory with sufficient optimization, reducing the area footprint in terms of board design.

Next to the limited resources, power efficiency plays a key role in the design of neural networks for spacecraft applications. Satellites have a limited power supply, especially when speaking about the Smallsats that could be early adopters of such AI systems. The power budget for such solution would be limited to a few Watts, to not impact the overall power budget of the satellite. Moreover, Thermal dissipation poses also another challenge, rendering *throughput/Watt* a much more important metric than total throughput.

In general, it is important to notice that the processing requirements in terms of latency and throughput are of second degree of importance, as long as they do not impact other critical applications on the spacecraft. For anomaly detection, the rates of telemetries are relatively low, and the latency is of the detection is mainly limited by the response of the OBC handling the fault than the anomaly detection calculation itself. For payload applications, the criticality of throughput depends on the mission types. Continuous downstream missions have critical throughput due to the bandwidth limitation, a relatively slow data rate, which could be met by the system requirements defined and the processing platforms envisioned. The processing latency would therefore be shadowed by the intrinsic limit of the downstream. This type of application is considered as the single stream or multi stream mode in the following benchmark description. Eventual downstream missions involve a limited set of downstream locations in specific points of the orbit, meaning that post-processing of the data will occur by fetching data from the long-term storage memory before the transmission window and not in real-time. This type of application is considered as the offline or batch mode in the following benchmark description.

Fault mitigation is another critical challenge for ML applications in space. Due to the processing requirements and tool availability, the best inference options are, in most cases, COTS devices and tools, which are sensitive to SEU[6] effects or, in the worst case, to failures affecting the reliability over the mission time of the equipment. Reliability issues of hardware platforms like GPUs or possible latch-up effects on MPSoC[7] are certainly a concern in future missions but are difficult to evaluate in the frame of the benchmark. Resistance to non-destructive radiation effects can instead be taken into account to compare different hardware platforms or even algorithms. On one side, FPGAs offer multiple options to reduce the effect of SEUs, while Fault-Aware Training allows users to train their models to include a certain resistance to bit upsets [14]. It is an open debate how relevant SEUs are in ML applications as on one side the algorithms are intrinsically quite resistant to bitflips and on the other side do not ensure a 100% accuracy in all cases, meaning that applications need to take into account false positives or negatives anyways. Nevertheless, mitigation techniques have proven to increase accuracy by a significant amount and are therefore paramount for many applications [13].

Another issue is the verification of AI driven applications. While commercial and agency missions have already proven the use of tool aided autocoding, issues relative to operating systems [19] and functional verification of ML algorithms are still a problem to be answered. These evaluations are out of scope of the benchmark, but submitters[8] will be able to highlight for example the use of real time operating systems when comparing to implementations based on Linux.

4. AI PROCESSING PLATFORMS IN SPACE

To select metrics that are on one side significant for the industry and on the other side allow for a comparison of a wide variety of submission platforms, an analysis of target processing platforms is performed. In this analysis, we enumerate the various families of processing platforms and discuss their capabilities and the scope of target applications.

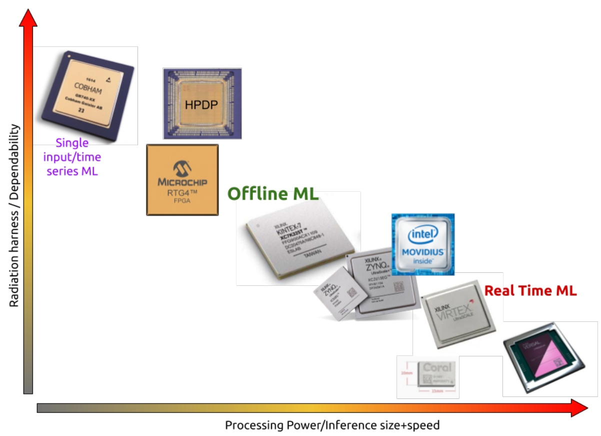

Figure 2: Target AI processing platforms for space, scattered in a spectrum of various processing power vs. degree of space qualification.

CPUs are one of the simpler options to implement machine learning algorithms. Due to the limitations in overall processing power and actual power efficiency of current Space Qualified CPUs, these are relegated to time series analysis applications, like anomaly detection. The next generation of system on chips for OBCs will involve multicore processors with much higher MFLOPS (e.g., DAHLIA or LEON4); therefore, larger neural networks could also be employed on such processing platforms. The inherent problem in such platforms is the computational efficiency when performing such complex calculations. Consequently, the power efficiency of embedded neural networks, even in the best cases, is x10 times lower

[6] Single-Event Upset

[7] **M**ulti**P**rocessor **S**ystem on a **C**hip

[8] similar to the structure of MLPerf, submitters develop and make a submission of the benchmark.

than a GPU or FPGA solution. Therefore, while implementing time series analysis with a limited scope is not a problem, the limitation of such a solution lies in the future scaling of the ML Anomaly detection products to include more and more telemetry evaluations.

FPGAs have been typically complex targets to implement Neural Networks due to the huge size of the design space and the high degree of manual optimization required to fully implement the network in programmable logic. Current developments in Design Tools from FPGA vendors, coupled with the advent of SoCs that combine the power of multicore-processing with FPGA-based hardware accelerators on the same board, enables usesrs to infer Convolutional Neural Networks on FPGAs suitable for space. Among the FPGA vendors, Xilinx offers the most mature in terms of tools for ML inference, especially with FINN (HLS based) and Vitis (DPU[9] IP based). FPGAs allow users to design tailored low-power machine learning applications that are compatible with equipment for satellite platforms, with the current generation being competitive even to Nvidia GPUs in terms of TOPS/W. The KU060[10] is already offered in an SEL[11] immune radiation-tolerant version, which could be envisioned as a coprocessor, especially targetting payload processors. Latch-up immune FPGAs from the Microchip portfolio have started to offer options for neural network inference with their Vectorblox[12] tool on the Polarfire[13], with the option of a softcore RISC-V processor.

SoC, MPSoC, ACAPs normally include both an FPGA and multiple fabric CPU cores, therefore allowing for increased flexibility when designing an application. Commercial or military SoC have shown acceptable single upset rates for non-critical applications, like payload processing, and latch-up effects have been characterized precisely, allowing necessary design precautions to ensure reliability using such integrated circuits. From the Xilinx portfolio the Zynq US+[14] is a promising solution for Smallsats, as well as being similar in terms of architecture to the NG-Ultra[15], and therefore representative of a future ITAR[16]-free processing solutions. Moreover, the next ACAP[17] generation, with the Versal[18], is expected to be qualified for space in the future and would be compatible in terms of tools, offering even higher performance. This class of processing chips can be kept in the same class as FPGAs in terms of implementation, considering that softcore processors can be introduced in programmable logic with softcore processors as described previously. The additional risk of SEFIs or other complex faults on the system, as well as a difficult radiation characterization, is compensated by a higher clock rate of the cores.

GPUs and VPUs from terrestrial applications, driven by CubeSat missions and the success of commercial Smallsat missions where reliability requirements are not driving the design, have been used more and more for image processing and AI. While sharing the framework with commercial solutions is clearly an advantage for the development time and complexity of an AI implementation on such platforms, the questions on reliability and availability restrict the use cases of such processing platforms. In fact, the use of terrestrial GPUs is highly questioned, given the high failure rate shown in radiation testing [10]. Smaller GPUs or VPUs, coming from IoT or Embedded commercial markets, like the Nvidia Jetson or Intel Myriad, have shown some promising results for shorter missions [11].

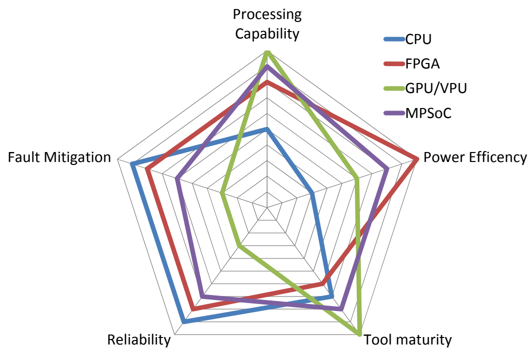

Figure 3: Quantitative comparison of processing various platforms

To sum up, in Figure 3 we illustrate a qualitative comparison the advantages and disadvantages of each platform. Finally, an extension of the benchmark to existing radiation tolerant array processors, like HPDP[19], or other promising architectures, like Kalray multicores, will be taken into account throughout the activity. It is to be noted that these dedicated ASICs might be difficult to benchmark using the use cases as the other platforms, and might require separate special categories.

5. ON-BOARD PROCESSING BENCHMARK DESIGN

In the previous sections, the main issues and needs of the space industry to support AI applications have been defined. Based on this, the benchmark should have a set of algorithms to be used for comparing of various processing platforms. The algorithms need to be representative of the workloads of each use case but allow for different categories in the design options space that are realistic for such applications. The parameters of each benchmark category need to be selected to match the necessary requirements of each use case. Another additional com-

[9] **D**eep Learning **P**rocessor **U**nit
[10] KU060 is a Xilinx Kintex product based on UltraScale architecture.
[11] **S**ingle-**E**vent **L**atch-up
[12] https://www.microsemi.com/product-directory/dev-tools/5597-vectorblox-ai
[13] Microchip PolarFire FPGAs
[14] A Xilinx system that features a UltraScale+MPSoC architecture
[15] FPGA-based MPSoC solution of NanoXplore
[16] International Traffic in Arms Regulations
[17] Xilinx **A**daptive **C**ompute **A**cceleration **P**latform
[18] Versal is Xilinx ACAP-based architecture
[19] High Performance Data Processor of ESA

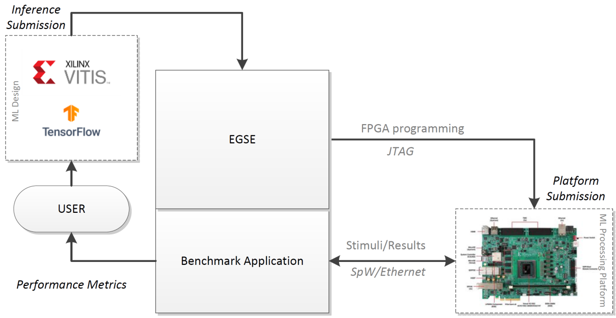

Figure 4: Illustration of benchmark submission example using Vitis AI framework.

plexity is that the benchmark needs to be able to demonstrate both the hardware and software capabilities of each platform. Moreover, given the variety of tools for inference, the benchmark needs to give an indication of how to perform the porting to the target hardware. **Closed Submissions** will *clearly* define which optimizations are allowed to ensure a fair comparison between different platforms. For example, quantization can be limited to 8-bit integers in a closed submission, ensuring that a realistic value is used and that most hardware can comply with this requirement. In the same way, submissions can be limited to single-threaded implementations to avoid variance and difference between processors and operating systems. Closed submissions will require a detailed analysis of each inference tool, to ensure that comparable results among various platforms are always achieved. Another important point regarding closed submissions is that they can give a realistic expectation on the design effort required for the implementation. **Open Submissions** will instead allow users to optimize their implementation freely, to showcase the possibly optimal performance that can be achieved with each processing platform. While such submissions could potentially make submissions difficult to compare, they will be useful to prove the feasibility of certain use cases.

Figure 4 shows an example of submission using a Xilinx board to clarify the benchmarking workflow. In this case, the model is designed and trained in Tensorflow. Then, the quantization is performed using Vitis AI framework to create an inference model for a (in this example) Versal ACAP. The benchmark shall then stimulated the model using the validation dataset and evaluate the performance. Note that goal of the study is to create a benchmark for ML platforms, focusing on the hardware aspect and the so-called *(hardware platform submissions)*. In the first place, the benchmark enables comparison among hardware options for ML, but users will be able to use the open submissions on a certain platform also to evaluate various algorithm models, which might be tailored to a specific architecture of the processing unit. This means that during the design possible *(inference model submissions)* need to be taken into account.

5.1. Benchmark Categories

The first proposed specifications for the benchmark are presented in (and not limited to) Table 1. The focus lies on neural networks used in Earth Observation tasks, as they are the most computationally intensive and share similar processing requirements as visual-based navigation algorithms. Nevertheless, the benchmark could be extended to include time series analysis or anomaly detection in the future.

Use Case	Model	Dataset
Image Classification (Heavy)	Resnet50 Multilabel	HyRank $\geq$384x384
Image Classification (Light)	MobileNet-v1 Singlelabel	BigEarthNet 224x224
Object Detection (Heavy)	UNet ResNet50	Airbus Ship $\geq$384x384
Object Detection (Light)	UNet Tiny YOLOv3	Airbus Ship 224x224
Hyperspectral ($220 \geq f \geq 13$)	3d UNet	Indian Pine fx144x144
Anomaly Detection (Heavy)	tCNN,Wavenet*	multi-input timeseries*
Anomaly Detection (Light)	GAN,RNN*	single-input timeseries*

Table 1: Preliminary specification of benchmark models, datasets and associated use cases.
*Note that anomaly Detection is introduced as a future extension of the benchmark and not discussed further.

In the current phase of the activity, we are considering a wide range of datasets and models to evaluate to define the most promising subset for the benchmark. The models selected need to not only be representative of real use case scenarios but also be supported by the widest range of inference tools. For this reason, the focus lies currently on the state-of-the-art for neural networks like ResNet, MobileNet, and YOLO. While there are plenty of freely available images from the Copernicus programme, the selection of dataset is challenging given the limited amount of labeled *"machine-learning-ready"* images. The MLAB project has the goal of analyzing datasets like labeled ships [3], coastlines [17], or large classification datasets like BigEarthNet [1]. The dataset should be openly available, ideally with an open-source license that allows for easy access. Given the issues of certain datasets (e.g., inaccurate labeling, data augmentation), possible adaptations of the dataset are not excluded if necessary. In this case, the general approach is to describe the necessary pre-processing steps within the benchmark. Table 1 shows the image sizes selected for the initial submissions. A "light" model focused on higher FPS and low accuracy, used to gather information for satellite decision, but not necessary to process actual downlink data, can be benchmarked on small images (224x224) as a standard for most applications. In an application scenario, these images would either come from a secondary sensor or be produced in a preprocessing step, e.g., cutting and binning the data. This ensures training with reasonable batch sizes and fewer issues in terms of embedded memory on the inference platform. A ”heavy” model would benefit from greater image sizes in the benchmark, in accordance to realistic scenarios for imaging sensors in which less binning can be performed. Currently, a tradeoff in terms of reasonable size for this ”heavy” use case is undergoing, with the Airbus Ship Detection dataset supporting image sizes up to 768x678. Commercial datasets, like COCO [4], support sizes of even 1200x1200 images for data center applications, which might be unrealistic on many embedded processing platforms and have to be analyzed further.

5.2. Benchmark Metrics

We are considering a number of performance metrics to characterize various platforms and solutions. These include various metrics such as accuracy, throughput, power consumption, tailed-latency, resource utilization, memory footprint, and fault-mitigation capabilities. However, in the context of on-board data processing, the accuracy, throughput, and power consumption are first-class citizens and specify minimum requirements for the submissions. Our benchmarks specify minimum acceptable thresholds for these metrics, and Table 2 illustrates a first proposal for such thresholds. For closed submissions, given the fact that accuracy, throughput, and power are correlated in a Pareto optimal design, the most relevant quality metric for comparison needs to be selected based based on each use case. The other metrics will be fixed to a value that is relevant to the use case. The alternative would be to fix all metrics in the closed submission and just see how much better each submission is than the reference. The disadvantage is that submissions might have difficult to compare results (e.g. optimized for power vs optimized for throughput).

For classification use case, the accuracy metric can be specified using Top-5, and for object detection, we rely on IoU[20] metric. Throughput is measured as the overall *frames per second* (FPS), which also includes the post-process time of the images as well as the actual inference on the FPGA chip. These throughput thresholds specify the minimum requirements and in many cases, are sufficient for the practical use case of Earth observation. Moreover, we are proposing power corridors based on the available power budged to the on-board inference engine.

In addition to the metrics discussed above, various metrics are considered as part of open submissions. For example, more aggressive activation quantization and model pruning can be considered as open submissions. Moreover, multi-threading on the CPU of MPSoC systems is another metric that has a direct impact on the overall performance (e.g., on throughput) and could be considered as part of an open submission. Open submission will allow to showcase the full potential of certain platforms with design constraints and make a difficult comparison with other processing options.

[20] Intersection over Union

Use Case	Accuracy	Throughput	Power
Image Class. (Heavy)	$Quality\ Metric$	5 FPS	10W
Image Class. (Light)	90%	10 FPS	$Quality\ Metric$
Obj. Detection (Heavy)	85%	$Quality\ Metric$	10W
Obj.Detection (Light)	80%	10 FPS	$Quality\ Metric$

Table 2: Preliminary specification of performance requirements for submissions. Further use cases submission options will be defined in the frame of the activity.

6. INFERENCE SUBMISSIONS

In this section, we introduce an implementation of the inference benchmark, and our primary evaluations. In order to illustrate our benchmark, we a compare an open and a closed submission with respect to multi-threading.

6.1. Specification of the Target System and Implementation

We rely on automatic and high-level model deployment workflow, i.e., Vitis AI as a fast path way to deployment. We target Xilinx ZCU102[21] system, which is an MPSoC system with a quad-core Arm Cortex-A53 processor, and a 16nm FinFET+ FPGA fabric, which is a reasonable testing platform for our benchmark. our inference scripts are implemented in Python and use the XIR[22] and VART[23] for the execution of inference. We use the board image of version 2020.2, and in all the experiments we use the Vitis AI library version 1.3.1.

6.2. Models and Dataset

For the experiments of this paper, we consider the deployment of a segmentation model for ship detection as an object detection use case. For this task, we used a *Unet-Resnet*-based segmentation model that was developed in the context of Airbus Ship Detection contest[24] which is based on the segmentation model pool developed by Yakubovskiy [9] in Keras and Tensorflow 1. We adapt this model and refactor the scripts based on the workflow introduced in Vitis Tutorials[25]. Moreover, we integrated the Unet implementation of Xilinx

[21] Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit

[22] Xilinx Intermediate Representation

[23] Vitis AI Runtime

[24] `https://www.kaggle.com/c/airbus-ship-detection`, `https://www.kaggle.com/awater1223/unet-resnet34-for-ships`

[25] `https://github.com/Xilinx/Vitis-Tutorials/blob/master/Machine_Learning/Design_Tutorials/05-Keras_FCN8_UNET_segmentation/files/code/config/unet.py`

in Keras which consist of layers that are all supported by Vitis AI for deployment on FPGA. With this implementation, we train and deploy segmentation models with various deep-network backbones, e.g., Unet-ResNet50, Unet-MobileNet-v2, and Unet-Inception-ResNet-V2, as well, as various image sizes, e.g., 224x224, and 384x384, quantization bits, e.g., 8 bits, and calibration configurations.

For the evaluation of the deployed models, we consider various model architectures and compare the overall throughput in Frames per Second (FPS). We consider one a closed submission in which the SoC only supports single threaded inference invocations and an open submission in which multi-threading is enabled.

6.3. Submission Preliminary Results

Table 3 presents a summary of preliminary results of our experiment using our benchmark prototype for the object detection use case, which will be tailored as benchmark submissions and used as an example. In Table 3, we observe an expected proportional throughput reduction with the increase of image frame sizes which is consistent among all network architectures and threading configurations. We further observe an expected increase in the throughput with the additional threads (referred to as *THD* in Table 3). This observation highlights the difference between a close and open submission with respect to threading: The open submission allows for achieving higher throughput by exploiting the threading capability of the CPU and operating system, while this can be forbidden in a closed submission. Consequently, the open submission allows for further optimizations for better utilization of the compute resources: This effect (increase in the throughput with the additional threads) is expected as using Vitis AI, the ZCU102 board is configured with 3 DPU units, and increasing the number of threads to request more inference queries simultaneously fills in the latency gaps and results in higher dynamic utilization of FPGA resources. The last observation is the comparison of various network architectures: while we are covering a wide range of network architectures (from the input size and network architecture point of view), we still cannot properly compare the networks. In other words, the network complexity and size are not yet reflected in the throughput. The reason for this is rooted in the currently limited support of ‘padding‘ layers in Vitis AI. This results in *partitioning* of computation graph into smaller *subgraphs* that are partly executed on the CPU and partly on the FPGA. Consequently, our current prototype is not an optimal implementation, and it suffers from the extra overhead of data flow between CPU and FPGA, and execution of the padding layers on CPU. This overhead is different for each individual network as each of them has a specific graph partitioning that renders throughput that are not in accord with the network size and complexity. However, this problem would be resolved as Vitis AI gets more mature. In that case, our benchmark would be capable of characterizing the effects of network architecture on the performance metrics. That being said, we also need to note that the current throughput values lie in a reasonable range for space applications. In the future, and using the MLAB benchmark, we perform similar analyses in combination with the other performance metrics, e.g., energy consumption and inference accuracy, to provide realistic benchmark specifications.

6.4. Future Hardware Submissions

In the frame of this activity or parallel activities that could support MLAB the following hardware submissions are envisioned:

- ZCU102 for Object Detection, with Vitis AI, FINN or Matlab AI toolbox
- Versal for Object Detection, with Vitis AI
- KU060 for Object Detection, with FINN
- Polarfire for Object Detection, with Vectorblox or Matlab AI toolbox
- ZCU102 for Anomaly Detection, with different frameworks
- Polarfire for Anomaly Detection, with different frameworks

Other submissions in terms of tools or hardware could be included as part of the benchmark demonstration.

7. CONCLUSIONS

This paper presented the state-of-the-art on-board ML tools, frameworks, and hardware platforms that target spacecrafts and satellites, demonstrating, on one side, the importance of AI in future missions and on the other challenges involved. We explained the need for developing benchmarks, with the goal of comparing hardware in the first place and potentially create a baseline of datasets and models to be used. The focus on neural networks for Earth Observation is justified, and different hardware options are described, with MPSoC being a promising option for such use case. The primary specifications for the machine learning application benchmark are described, with a first structure proposed for such applications. Preliminary results show the feasibility of implementing ML applications (Ship Detection) for space, using simple and automatic hardware design workflows (Vitis AI) to deploy AI algorithms on hardware that meets the requirements of future missions.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the support and guidance provided by the ESA technical officer Gianluca Furano under the scope of the MLAB contract.

Architecture	Unet		Unet-MobileNetv2		Unet-ResNet50		Unet-InceptionResNetv2	
Size	160x160	224x224	160x160	224x224	160x160	224x224	160x160	224x224
Throughput - 1 THD.	21.40	11.48	20.89	12.00	19.64	10.53	19.49	14.56
Throughput - 8 THDs.	45.80	25.61	47.70	27.23	36.34	23.42	41.32	29.83

Table 3: Throughput (in FPS) of Semantic segmentation on Air Ship Detection dataset on ZCU102. The models are trained with 100 Epoches, and then quantized with 8-bit weights and activations.

REFERENCES

[1] `http://bigearth.net/`

[2] `https://spacenet.ai/datasets/`

[3] `https://sandbox.intelligence-airbusds.com/web/`

[4] `https://cocodataset.org`

[5] `https://github.com/Xilinx/finn-hlslib`

[6] `https://github.com/Xilinx/Vitis-AI`

[7] `https://github.com/fastmachinelearning/hls4ml`

[8] `https://github.com/Microchip-Vectorblox/VectorBlox-SDK`

[9] `https://github.com/qubvel/segmentation_models`

[10] Proton Testing of AMD e9173 GPU, NASA Goddard Space Flight Center, Technical Rep. No. GSFC-E-DAA-TN72682, 2019

[11] W. S. Slater, N. P. Tiwari, T. M. Lovelly and J. K. Mee, ”Total Ionizing Dose Radiation Testing of NVIDIA Jetson Nano GPUs,” 2020 IEEE High Performance Extreme Computing Conference (HPEC), 2020, pp. 1-3, doi: 10.1109/HPEC43674.2020.9286222.

[12] Wei X. et al., FPGA-Based Hybrid-Type Implementation of Quantized Neural Networks for Remote Sensing Applications, 2019

[13] Zahid, Ussama et al., FAT: Training Neural Networks for Reliable Inference Under Hardware Faults. 2020 IEEE International Test Conference (ITC) (2020): 1-10.

[14] G. Gambardella et al., ”Efficient Error-Tolerant Quantized Neural Network Accelerators,” 2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2019, pp. 1-6, doi: 10.1109/DFT.2019.8875314.

[15] F. Libano et al., ”Selective Hardening for Neural Networks in FPGAs,” in IEEE Transactions on Nuclear Science, vol. 66, no. 1, pp. 216-222, Jan. 2019, doi: 10.1109/TNS.2018.2884460.

[16] J. Manning et al., Machine-learning space applications on Smallsat platforms with tensorflow, in Proceedings of the 32nd Annual AIAA/USU Conference on Small Satellites, Logan, UT, USA, 2018, pp. 49

[17] Ting Yang, Shenlu Jiang, Zhonghua Hong, Yun Zhang, Yanling Han, Ruyan Zhou, Jing Wang, Shuhu Yang, Xiaohua Tong & Tae-yong Kuc (2020) Sea-Land Segmentation Using Deep Learning Techniques for Landsat-8 OLI Imagery, Marine Geodesy, 43:2, 105-133, DOI: 10.1080/01490419.2020.1713266

[18] V. J. Reddi et al., ”MLPerf Inference Benchmark,” 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), 2020, pp. 446-459, doi: 10.1109/ISCA45697.2020.00045.

[19] H. Leppinen, ”Current use of linux in spacecraft flight software,” in IEEE Aerospace and Electronic Systems Magazine, vol. 32, no. 10, pp. 4-13, October 2017, doi: 10.1109/MAES.2017.160182.

[20] M/ Esposito, ”In-orbit demonstration of artificial intelligence applied to hyperspectral and thermal sensing from space”, in CubeSats and Smallsats for remote sensing III vol. 11131

[21] Furano, Gianluca & Menicucci, Alessandra. (2018). Roadmap for On-Board Processing and Data Handling Systems in Space. 10.1007/978-3-319-54422-9_10.