Machine Learning Application Benchmark

Invited Paper

Andreas Koch*
andreas.c.koch@airbus.com
Airbus Defence & Space GmbH
Munich, Germany

Michael Petry†
michael.petry@airbus.com
Airbus Defence & Space GmbH
Munich, Germany

Max Ghiglione
max.ghiglione@esa.int
European Space Agency
Noordwijk, The Netherlands

Amir Raoofy‡
amir.raoofy@tum.de
Technical University of Munich
Munich, Germany

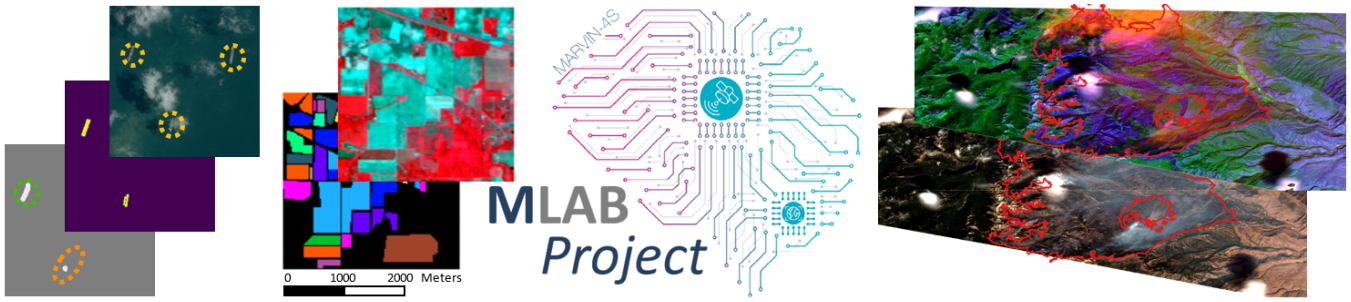
Gabriel Dax
gabriel.dax@tum.de
Technical University of Munich
Munich, Germany

Gianluca Furano
gianluca.furano@esa.int
European Space Agency
Noordwijk, The Netherlands

Martin Werner
martin.werner@tum.de
Technical University of Munich
Munich, Germany

Carsten Trinitis
carsten.trinitis@tum.de
Technical University of Munich
Munich, Germany

Martin Langer
martin.langer@ororatech.de
Orbital Oracle Technologies
Munich, Germany



ABSTRACT

This paper presents 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 need for a benchmark dedicated to machine learning applications in spacecraft is explained, and examples of applications are described including their design challenges. Then the benchmark design is presented, including the rules of the metrics, guidelines and scenarios for references. These scenarios include a description of the reference workloads that have been selected during the activity as representative for spacecraft applications. Lastly, the submission concept is introduced.

CCS CONCEPTS

• **Hardware** → **Emerging architectures**; • **Computing methodologies** → *Artificial intelligence*; **Computer vision tasks**.

KEYWORDS

Machine learning, neural networks, benchmark, FPGA, datasets, power consumption

*Also with Technical University of Munich.
†Also with Technical University of Munich.
‡Also with Leibniz Supercomputing Centre.

CF '23, May 9–11, 2023, Bologna, Italy

ACM ISBN 979-8-4007-0140-5/23/05.
https://doi.org/10.1145/3587135.3592769

ACM Reference Format:
Andreas Koch, Michael Petry, Max Ghiglione, Amir Raoofy, Gabriel Dax, Gianluca Furano, Martin Werner, Carsten Trinitis, and Martin Langer. 2023. Machine Learning Application Benchmark: Invited Paper. In *20th ACM International Conference on Computing Frontiers (CF '23), May 9–11, 2023, Bologna, Italy*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3587135.3592769

1 INTRODUCTION

Satellites and other spacecrafts are an important tool for many applications routinely used today including safe and reliable communication, Earth observation, global navigation, and more. Due to the remote operation and difficulty for ground operators to quickly react and recover on-board failures due to limited visibility periods, autonomy is paramount. For this purpose, Machine Learning (ML) is increasingly used in space demonstration missions, like e.g. ESA's phi-sat, paving the way towards its mainstream use in Smallsats and in large institutional projects. This trend is fueled by the use of Commercial-Off-The-Shelf (COTS) solutions and the improvement of tools for ML deployment on radiation-tolerant processing units[1]. For successful deployment and use of ML in space, however, we are facing three main challenges: 1) on-board spacecraft hardware has limited processing capabilities, algorithms are required 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 are limited in terms of open accessibility and re-usability for space missions, as data is either proprietary and label availability is very limited. As a result, there is a clear need for higher transparency and comparability of ML implementations in the spacecraft domain, especially taking the space environment and representative applications into account. For this reason, we introduce the ML benchmark **Machine Learning Application Benchmark** (MLAB), focused on on-board spacecraft applications. On top of previous work discussing the benchmark concept[8], it shall provide use cases, reference datasets, reference models, reference hardware implementations and guidelines for submission of concrete use cases to the benchmark, which will all be made available on the MLAB submission platform[1]. The remainder of the paper is structured as follows: In section 2, we give an overview of the benchmark. Section 3 defines rules to scenarios, submissions and models. Reference workloads are outlined in section 4 and details of the benchmarking scripts are presented in section 5. Lastly, the submission process is specified in section 6.

2 BENCHMARK OVERVIEW

For the benchmark, first, a particular space use case is defined in terms of a scenario containing a description of the deployment (batch, streaming, ad-hoc), the ML model candidate to benchmark, and a single dataset to underpin the benchmark with workloads. In this way, typically for a given scenario many benchmark instances need to be run to explore the design space. In addition, each benchmark specifies a set of measurable requirements constraining the design space. In this way, a benchmark can either fully fail (e.g., placement is impossible), fail to adhere to quality attributes (e.g., performance too low) or succeed within the constraints. In the project, we are currently considering various tasks associated with ML in space. The main focus is on vision tasks, e.g. classification and object detection, as this area might give the highest gain in terms of saving communication cost and energy costs on-board. However, we are considering anomaly detection tasks as well and the framework is deliberately open to other types of AI. In fact, it is not even limited to Deep Neural Networks, but can be used much more generic for benchmarking spacecraft hardware. The targets for MLAB mostly consist of implementations utilizing Field Programmable Gate Arrays (FPGAs), since they can be programmed to contain specialized processors suitable for hardware acceleration of ML models. In order to immediately put new submissions into context, we have gathered a database of datasets, models, and benchmark results such that novel hardware aspects such as a new system under test, a new hardware implementation of an ML algorithm, or any other novel approach can be analyzed step by step by adding required changes to established benchmarks. We gather and organize a database of benchmark records that are successfully prepared using a submission system. Furthermore, a submission system is proposed as the user interfaces for the benchmark developers to provide the results of their benchmarks. Therefore, this submission system includes a simple database for record and metadata keeping as well as checker scripts that aim at assessing the satisfaction of performance requirements of a benchmark instance.

[1]https://github.com/mwernerds/mlab-benchmark

2.1 Benchmark Metrics

By analyzing the requirements of multiple scenarios and workflows and relying on diverse model implementations on an FPGA, we collected the following metrics for the benchmark:
Power consumption:

- Average power consumption (`Watt`).
- Peak power consumption (`Watt`)

Throughput:

- Frames per Second (`FPS`)
- Pixels per Second (`PPS`)

Energy efficiency:

- Frame/Watt
- Pixels/Watt

Accuracy:

- Accuracy (Top 1, Top N)
- IoU
- mAP

Each use case in the specification will provide the requirements for the above metrics. Note that as frame size might change depending on the use case, units like pixel/watt are also defined.

2.2 Benchmark concept

Existing benchmarks, like MLPerf[9], rely on the use of four separate components for the development of the benchmark. These components are System Under Test (SUT), the Load Generator (LoadGen), a dataset, and a metric evaluation suite, and help to generate reproducible inference experiments for benchmarking. SUT interacts with Load Generator and Dataset components to generate consistent inference traffic for different platforms. The Load Generator is used to develop consistent and comparable inference workloads. This includes the development of various types of queries (e.g., multi-stream inference queries). The Metric Evaluation component interacts with the Load Generator and is used to derive consistent performance metrics from the execution of benchmarks among various platforms.
The MLAB benchmark builds upon this concept and applies it to on-board spacecraft applications. The system under test should

incorporate space ready hardware and at least provide a description as to how flight-ready its software is.

3 BENCHMARK RULES

In the spirit of the state of the art MLPerf benchmarking method, we provide a set of definitions and restrictions, which help to specify the benchmarks:

3.1 Definition

- `Sample`: is the unit of data on which inference is run. E.g., a single image.
- `Query`: is a set of N samples that are issued to an inference system together. For example, a single query can contain 8 images.
- `Quality`: refers to a model's ability to produce "correct" outputs. This ability is concretely defined for each particular benchmark instance.
- `SUT`: consists of a defined set of hardware and software resources that will be used for for performance/quality measurements. The hardware resources may include processors, accelerators, memories, disks, and interconnect. The software resources may include an operating system, compilers, libraries, and drivers that significantly influences the running time of a benchmark. In addition, we are also including the data generation and dataset as part of SUT.
- `Reference implementation`: is a specific implementation of a benchmark provided by Airbus . The reference implementation is the canonical implementation of a benchmark.
- `Run`: is a complete execution of a benchmark implementation on a system consisting of a complete set of inference queries, including data pre- and post-processing, throughput, and power consumption requirements as well as a quality requirement in accordance with a use case.

3.2 Restrictions

Since we are mainly targeting accelerated platforms, we define various restrictions in order to measure more directly the device capabilities as opposed to performance features related to system implementation strategies:

- Caching of data on the accelerator is not allowed
- Throughput is measured for the end-to-end data pipeline
- Randomness and non-determinism are restricted and only allowed for the order of floating-point operations, traversal of the input streams, and rounding effects.
- Reproducibility is mandatory including the restriction on caching of data on the FPGAs

The final repository will include the full set of rules.

3.3 Benchmarking Scenarios and Use Cases

To cover the realistic inference scenarios for space applications, we define two scenarios as described in the table 1. These scenarios aim for performance evaluation for various HW while considering feeding the input data to the accelerator with either batches or streaming. Each benchmark specifies a scenario to which the submissions should adhere. The following table provides the details of the two scenarios discussed.

In the development of the benchmark, we were mainly considering use cases in which a full FPGA is available as a payload processor. In this situation, computer vision tasks, especially CNNs, provide a good "challenge" to the system for which reasonable data is available. Hence, the MLAB benchmark currently has a focus on vision-related tasks, yet contains a few hints on other use cases. We aim to extend the set of use cases in future work. Table 2 lists the main use cases available in the benchmark as of now.

3.4 Open and close divisions

When providing the benchmark reference metrics with a custom submission, different options can be chosen to implement a design. The following options of submission to the benchmark have been defined and they specify whether change in either the model or hardware configuration is allowed:

- Hardware Open & Model Open
- Hardware Close & Model Open
- Hardware Open & Model Closed
- Hardware Closed & Model Closed

While all these cases have relevance, we mainly focus on cases where the benchmark is *closed* with respect to the *model*: Preprocessing, post-processing and the model have to be equivalent to the reference implementation. This allows more straightforward hardware evaluation with less configuration noise from the models side.

3.4.1 Open Hardware & Open Model. In this case, also called open submission, only the use case and the dataset is specified. The submitter can choose to showcase any kind of performance optimization, but can use the reference use cases as baseline for the design.

3.4.2 Open Hardware & Closed Model. In this case, also called closed model submission, the hardware and inference implementation (e.g. accelerator IP) is modified keeping the same neural network model. This allows showcasing the advantages of one processing architecture against the other, and is relevant in a system design scenario where for example the model is fixed because already optimized or coming from another company or team. In general this use of the benchmark is also relevant to benchmark the capabilities of hardware in terms of quantization optimization and similar.

3.4.3 Closed Hardware & Open Model. In this case, also called closed hardware submission, the model is modified keeping the same hardware platform. This demonstrates the advantages of one model or training method against the other, and is relevant in a system design scenario where the hardware is fixed.

3.4.4 Closed Hardware & Closed Model. In this case, also called closed submission, a particular hardware platform, as well as the framework, compiler, deployment, and runtime environment, are specified. This is necessary to be able to compare different hardware platforms in terms of neural network inference capabilites with fairness in the implementation.

Table 1: Query Scenarios

Scenario	Query Generation	Min. # of queries	Samples per query	Min. duration	Min. # of runs
Stream	Continuous data stream	1024 queries	1	120 seconds	10
Batch	Batch-wise data stream	1024 queries	1024	120 seconds	10

Table 2: MLAB use cases & References

#	Use case	Reference Model	Reference Dataset	Reference Training	Reference Inference
A	Anomaly Detection Light	LSTM	NASA Anomaly	Matlab	Matlab
B	Radio Classification	ResNet 1d	RadioML	Tensorflow	Vitis AI
C	Multispectral Object Detection	VGG	Fire Detection	Tensorflow	Vitis AI
D	Image Classification Heavy	DenseNet161	EuroSAT	Tensorflow	Vitis AI
E	Image Classification Light	MobileNet	Airbus Ship Small	Pytorch	FINN
F	Image Object Detection	YOLO	Airbus Airplane	Darknet	Vitis AI
G	Image Segmentation	ResNet-50	Airbus Ship Masks	Tensorflow	Vitis AI

3.5 Model Rules

For the closed model division, MLAB provides a reference implementation of each benchmark. The benchmark implementation must use a model that is equivalent, as defined in these rules, to the model used in the reference implementation. For the open model division, the benchmark implementation may use a different model to perform the same task. Retraining is allowed.

3.5.1 Quantization. For the closed model division, MLAB will provide trained weights and biases in fp32 format for both the reference and alternative implementations. In addition, a calibration dataset will be published for each model. Submitters may do arbitrary purely mathematical, reproducible quantization using only the calibration data and weight and bias tensors from the benchmark owner provided model to any numerical format that achieves the desired quality. The quantization method must be publicly described at a level where it could be reproduced. Calibration is allowed and must only use the calibration dataset provided by the benchmark owner. Submitters may choose to use only a subset of the calibration data set. Additionally, MLAB may provide an INT8 reference for some models to increase comparability of this widely-used quantization size related to the availablitiy of fast elements (e.g., multipliers, memory access, ...) based on this bit size or multiples thereof in modern FPGAs. Model weights and input activations are scaled per tensor, and must preserve the same shape modulo padding. Convolution layers are allowed to be in either NCHW or NHWC format. No other retraining is allowed. For the closed and open division, weights and biases must be initialized to the same values for each run, any quantization scheme is allowed that achieves the desired quality.

4 REFERENCE WORKLOADS

This section defines the content of a reference workload of a use case and presents an overview of reference datasets and models. Reference workloads serve the purpose to demonstrate a benchmark implementation and provide results for comparison. All reference implementations provided by Airbus Defence and Space GmbH will utilize the Xilinx Zynq Ultrascale+ MPSoc ZCU102 evaluation kit[2]. It allows a wide range of HW designs, while being a target platform for the hardware acceleration frameworks of Xilinx Vitis AI, Xilinx FINN and the Matlab Deep Learning HDL toolbox. Each use case has a reference workload, which consists of:

- *Reference model*, selected from literature
- *Reference dataset*, open source with pre-processing scripts
- *Reference training*, on one of the frameworks
- *Reference inference*, on one of the frameworks
- *Reference hardware*, all references on ZCU102

4.1 Use Case A: Anomaly Detection Light

Nominal values for a multivariate timeseries of satellite telemetry will be predicted[5]. All metrics for power consumption, throughput and energy efficiency are applicable. Accuracy will be measured in line with a binary classification task (nominal or anomalous) on every value of a sample.

- **Reference Dataset**: The NASA Anomaly dataset contains labeled telemetry from the Mars Science Laboratory rover, Curiosity (MSL) and the Soil Moisture Active Passive satellite (SMAP)[3]
- **Reference Model**: In line with [5], a LSTM model will be used, with hardware acceleration provided by the Matlab Deep Learning HDL toolbox
- **Target Metrics**: Accuracy > 80%, Power
- **Query Scenarios**: Batch

4.2 Use Case B: Radio Classification

This use case comprises the prediction of which modulation is used on a given signal, based on its vector-like representation of the IQ plane[7].

- **Reference Dataset**: Open RadioML Synthetic Benchmark Dataset[4]

[2]https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html
[3]https://github.com/khundman/telemanom
[4]https://github.com/radioML

- **Reference Model**: ResNet 1d model[2], which will be accelerated via the Vitis AI framework
- **Target Metrics**: Accuracy > 50%, Throughput
- **Query Scenarios**: Batch

4.3 Use Case C: Multispectral Object Detection

In the context of AI-assisted computation in space, object detection tasks have great importance, as they can potentially serve as one of the critical building blocks in many applications. For this use case the detection of wildfires based on optical images is proposed. Images contain landmass-area infested by fires, as well as smoke formations and pixel-level masks should delimit the fire area. Accuracy will be measured as a binary object detection task of fire / no fire, with any overlap of bounding box and segmentation label mask being enough for a true positive.

- **Reference Dataset**: OroraTech Wildfire Dataset containing 11347 optical images from Sentinel satellites, labeled on pixel level[5]
- **Reference Model**: VGG16 models have been shown to work for the task of wildfire detection[6]. The model predicts bounding boxes based on optical images. It was selected for its ease of use within Vitis AI
- **Target Metrics**: Accuracy > 90%, Throughput
- **Query Scenarios**: Batch

4.4 Use Case D: Image Classification Heavy

In this use case some "heavy" models are evaluated that reach high performance measures, but suffer from high resource utilization and complex training implying the need for large datasets.

- **Reference Dataset**: The EuroSAT dataset is composed of aerial image tiles showing varying land-use classes in RGB colors as well as in 13 multispectral bands with resolutions varying between 10m, 20m and 60m. It contains 27,000 georeferenced images from Sentintel 2, with a size of 256x256 pixels. Each image is associated with one out of ten classes[6]
- **Reference Model**: DenseNet161 model[4] considering only 3 out of 16 channels for every image in the EuroSAT dataset. This model was selected as a reference as it fits the image classification heavy use case due to its size, while still being suitable for acceleration with Vitis AI
- **Target Metrics**: Accuracy > 90%, Throughput
- **Query Scenarios**: Batch

4.5 Use Case E: Image Classification Light

This use case involves only a single class, meaning that the algorithm needs to decide whether a specific object is in the image or not. With the goal of a smaller memory footprint, a lower classification accuracy is acceptable.

- **Reference Dataset**: Part of the Airbus Ship Detection Challenge, adapted for a binary classification task[7]
- **Reference Model**: MobileNet model[3] with hardware acceleration planned via the FINN framework
- **Target Metrics**: Accuracy > 80%, Power
- **Query Scenarios**: Batch

4.6 Use Case F: Image Object Detection

As a second object detection use case, the detection of aircrafts based on optical images is chosen. Accuracy will be measured via average precision (AP).

- **Reference Dataset**: The Airbus Aircraft Detection dataset is a demonstration of a larger dataset created from Airbus HRS imagery[8]
- **Reference Model**: YOLO model[10] for airplane detection and on-board inference via Vitis AI
- **Target Metrics**: Accuracy > 90%, Throughput
- **Query Scenarios**: Batch

4.7 Use Case G: Image Segmentation

Image segmentation has the goal to cluster parts of an image together which belong to the same class. As one of the more demanding algorithms in terms of neural networks for image processing, it is picked to showcase the performance of ship segmentation on optical images. For measuring accuracy, we will rely on the intersection over union (IoU) metric.

- **Reference Dataset**: Airbus Ship Detection Challenge dataset with segmentation label masks[7]
- **Reference Model**: ResNet-50 model[2] containing a zero-padding layer, which constitutes an operation that is not supported by the Vitis AI DPU and needs to be accounted for during deployment
- **Target Metrics**: Accuracy > 60%, Power
- **Query Scenarios**: Batch

5 BENCHMARKING SCRIPTS FOR REFERENCE IMPLEMENTATIONS AND SUTS (ZCU102)

According to the rules set in the benchmarking section, the algorithms are tested for performance under different perspectives. The base dimensions for the benchmarking are:

- Accuracy
- Timing
- Power Consumption

From the dimensions above the following can be derived:

- Accelerator execution time
- Accelerator throughput
- Inference Energy
- Accelerator/Model energy efficiency

5.1 Accuracy

Whereas the computation of the accuracy metric for classification tasks is trivial, it is more complex for the image segmentation cases. The accuracy in the image segmentation cases is computed by averaging the IoU score across all the images composing the batch.

[5] https://ororatech.com/
[6] https://github.com/phelber/EuroSAT
[7] https://www.kaggle.com/c/airbus-ship-detection
[8] https://www.kaggle.com/datasets/airbusgeo/airbus-aircrafts-sample-dataset

5.2 Timing

The timing of the algorithm comprises the operations required to adapt the data for the algorithm, feed it to the accelerator, perform the computations in the accelerator and return the results of the operations.

5.3 Power Consumption

To measure the power consumption, the Maxim PM-Bus is sampled at constant intervals. The values sampled from the power-rails are grouped in the following categories:

- PL Power
- PS Power
- MGMT Power

Since all video drivers and codecs of the board are not utilized, the MGMT power reading shows only idle-state currents, which are negligible.

5.4 Accelerator execution time

The execution time of operations in the accelerator is obtained either by means of the VAI-Trace profiler, which is able to provide execution times down to operation level for DPU-compiled code, or by utilizing PL the power-consumption rising and setting flanks as delimiters. Vitis-AI allows for the first approach, the lack of a profiler for FINN only allows the second approach.

5.5 Accelerator throughput

The throughput of the accelerator is computed by dividing the accelerator execution time by the size in bit of the batch.

5.6 Inference Energy

The inference energy is computed by performing a trapezoidal numeric integral on the power-consumption profile. The limits of the numeric integral are the rising and setting flanks of the accelerator power consumption. The energy measurement contains the both the PS and PL power, as PS operations are needed for PL scheduling.

5.7 Accelerator/model energy efficiency

The accelerator energy efficiency, is computed for every model by dividing the inference energy by the size of the batch in bit. The latter gives insights about overheads, inner workings, and optimizations of the accelerator.

6 BENCHMARK SUBMISSION PROCEDURE

We define a submission as the preparation of code, scripts, measurements and summaries for a particular benchmark scenario and use case. Each submission is bound to a particular hardware platform, compiler and deployment software. Currently, the reference implementations are prepared for ZCU102 platform. However, we encourage the submission of other platforms and systems. Therefore we prepared an instruction to help developers preparing submissions. This instruction, includes the following steps:

- Check the existing scenarios and use cases
- Prepare reproducibility materials
- Structure the evaluation metrics
- Verify submission requirements
- Pull request and final submission

7 CONCLUSION

In this paper, we present the current status of the MLAB benchmark system for space applications. Based on a selected set of publicly available use cases for on-board processing, we derive a proper set of metrics and a benchmarking protocol tailored to on-board inference in space. The standardized workloads and performance metrics together with a growing range of documented use cases and reference implementations for real platforms provide system engineers with a framework to place evaluated decisions during system specification in terms of hardware selection, software framework benchmarking, and model design. In addition, the benchmark helps hardware designers to assess their hardware in the context of satellite-based workloads. Finally, it provides a means of understanding to the machine learning community how real-world application performance diverges from the current evaluation scheme only focusing on peak dataset performance in this community. To keep the benchmark in line with existing on-board processing applications, we continue working on improving the coverage and diversity of the benchmark use cases and are hoping that this activity will simplify the integration of machine learning aspects into future space missions.

REFERENCES

[1] Max Ghiglione and Vittorio Serra. 2022. Opportunities and Challenges of AI on Satellite Processing Units. In *Proceedings of the 19th ACM International Conference on Computing Frontiers* (Turin, Italy) *(CF '22)*. Association for Computing Machinery, New York, NY, USA, 221–224. https://doi.org/10.1145/3528416.3530985

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. (Dec 2015). https://doi.org/10.48550/arXiv.1512.03385 arXiv:1512.03385 [cs].

[3] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. (Apr 2017). https://doi.org/10.48550/arXiv.1704.04861 arXiv:1704.04861 [cs].

[4] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2018. Densely Connected Convolutional Networks. (Jan 2018). https://doi.org/10.48550/arXiv.1608.06993 arXiv:1608.06993 [cs].

[5] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. 2018. Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 387–395. https://doi.org/10.1145/3219819.3219845

[6] Wonjae Lee, Seonghyun Kim, Yong-Tae Lee, Hyun-Woo Lee, and Min Choi. 2017. Deep neural networks for wild fire detection with unmanned aerial vehicle. In *2017 IEEE International Conference on Consumer Electronics (ICCE)*. 252–253. https://doi.org/10.1109/ICCE.2017.7889305

[7] Timothy James O'Shea, Tamoghna Roy, and T. Charles Clancy. 2018. Over-the-Air Deep Learning Based Radio Signal Classification. *IEEE Journal of Selected Topics in Signal Processing* 12, 1 (2018), 168–179. https://doi.org/10.1109/JSTSP.2018.2797022

[8] Amir Raoofy, Gabriel Dax, Vittorio Serra, Max Ghiglione, Martin Werner, and Carsten Trinitis. 2022. Benchmarking and Feasibility Aspects of Machine Learning in Space Systems. In *Proceedings of the 19th ACM International Conference on Computing Frontiers* (Turin, Italy) *(CF '22)*. Association for Computing Machinery, New York, NY, USA, 225–226. https://doi.org/10.1145/3528416.3530986

[9] Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maximilien Breughe, Mark Charlebois, William Chou, Ramesh Chukka, Cody Coleman, Sam Davis, Pan Deng, Greg Diamos, Jared Duke, Dave Fick, J. Scott Gardner, Itay Hubara, Sachin Idgunji, Thomas B. Jablin, Jeff Jiao, Tom St. John, Pankaj Kanwar, David Lee, Jeffery Liao, Anton Lokhmotov, Francisco Massa, Peng Meng, Paulius Micikevicius, Colin Osborne, Gennady Pekhimenko, Arun Tejusve Raghunath Rajan, Dilip Sequeira, Ashish Sirasao, Fei Sun, Hanlin Tang, Michael Thomson, Frank Wei, Ephrem Wu, Lingjie Xu, Koichi Yamada, Bing Yu, George Yuan, Aaron Zhong,

Peizhao Zhang, and Yuchen Zhou. 2019. MLPerf Inference Benchmark. *CoRR* abs/1911.02549 (2019). arXiv:1911.02549 http://arxiv.org/abs/1911.02549

[10] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. (May 2016). https://doi.org/10.48550/arXiv.1506.02640 arXiv:1506.02640 [cs].