EC7

ML Benchmarking Environment Definition

Definition of target hardware platforms, frameworks and models

contact@confiance-ai.fr | www.confiance.ai

Specific information to be provided in ANR reports only

X	Progress report		Final report
From:	AAAA/MM/JJ	To:	AAAA/MM/JJ
Brief description of the project (reminder)			
Contractual description of the project:		~~nom du fichier~~	
Reference budget:		~~nom du fichier~~	

RECOMMANDATIONS

Strikethrough text is used to illustrate and explain the headings. It should be removed and replaced with non-strikethrough text. Similarly, this recommendations block should be removed, along with the statement "Specific information to be provided in ANR reports only", prior to dissemination of the document.

ML Benchmarking

Environment

Document reference: XXX

Contributors

	Name	Organisation	Role
Responsible for the deliverable	E. JENN	IRT Saint Exupéry	Action sheet leader
Scientific responsible	E. JENN	IRT Saint Exupéry	Action sheet leader
Co-authors	N. ABDERRAHMANE T. ALLOUCHE L. DANIEL F. FERESIN O. HLIMI E. JENN Ch. MARABOTTO F. THIANT	IRT Saint Exupery ATOS AzurIA AzurIA IRT Saint Exupery IRT Saint Exupery IRT Saint Exupery IRT SystemX	Research engineers

Document control

Revision	Date	Commentary	Author
1.0	2022/07/05	Creation	All co-authors

A. Introduction 6
A. Target platforms 9
A.1 Approach 9
A.2 Attributes 11
A.3 Selection 12
B. Frameworks 15
B.1 Approach 15
B.2 Attributes 15
B.3 Selection 16
C. Models 18
C.1 Approach 18
C.2 Attributes 20
C.3 Selection 20

I - Introduction

A. Introduction

The objective of the **ML benchmarking environment** (or **ML-bench**) is to provide the technical means and the guidance to support and optimize the deployment process of Machine Learning models.

For a given ML model, the ML-bench will provide means to (1) **select** a deployment framework and execution platform, (2) deploy the model on this environment, (3) evaluate and (4) optimize performances according to different strategies (using direct measurements on the actual hardware or via models), and (5) compare the performances achieved for different deployments (see Figure 1).

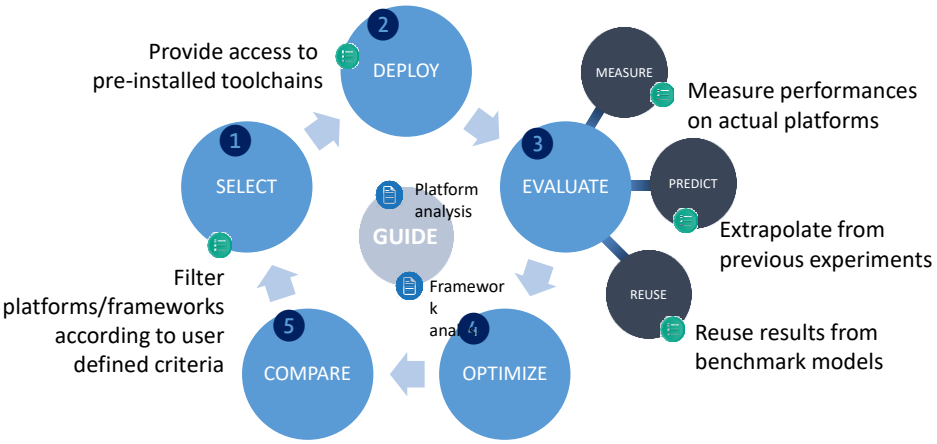


Figure 1. Objectives of the ML benchmarking environment

Even though the ML-bench does not directly contributes *per se* to the confidence on the system embedding ML, it plays an important indirect role since confidence and efficiency are tightly coupled. In particular, deploying a model on a resource-constrained target (or some specific targets such as FPGAs) may require model transformations such as quantization that represent a potentially significant alteration of "semantics" of the training model. Moreover, the choice of a GPU to ensure latency requirements may have a significant impact on temporal determinism or, at least, on the capability to demonstrate this temporal determinism. Finally, reducing the memory and/or CPU resources by selecting an appropriate deployment framework and/or target platform is a means to leave more room for various forms of temporal and spatial redundancies. For all these reasons, the ML-Bench is an integral and important component of the Confiance.ai environment.

The overall architecture of the ML-bench is depicted on Figure 2. This document presents successively its three main components:

- A set of **hardware targets** to which ML models can be deployed
- A set of **frameworks** to transform the ML models into representations executable on the targets
- A set of **selected models** to provide reference results.

Besides providing reference results, the deployment of the reference models will validate the benchmark environment detailed in the next figure. The performances figures obtained on these configurations will be crosschecked against the selection tool that will be developed as help for the external user to choose the best combination of AI model and HW targets.

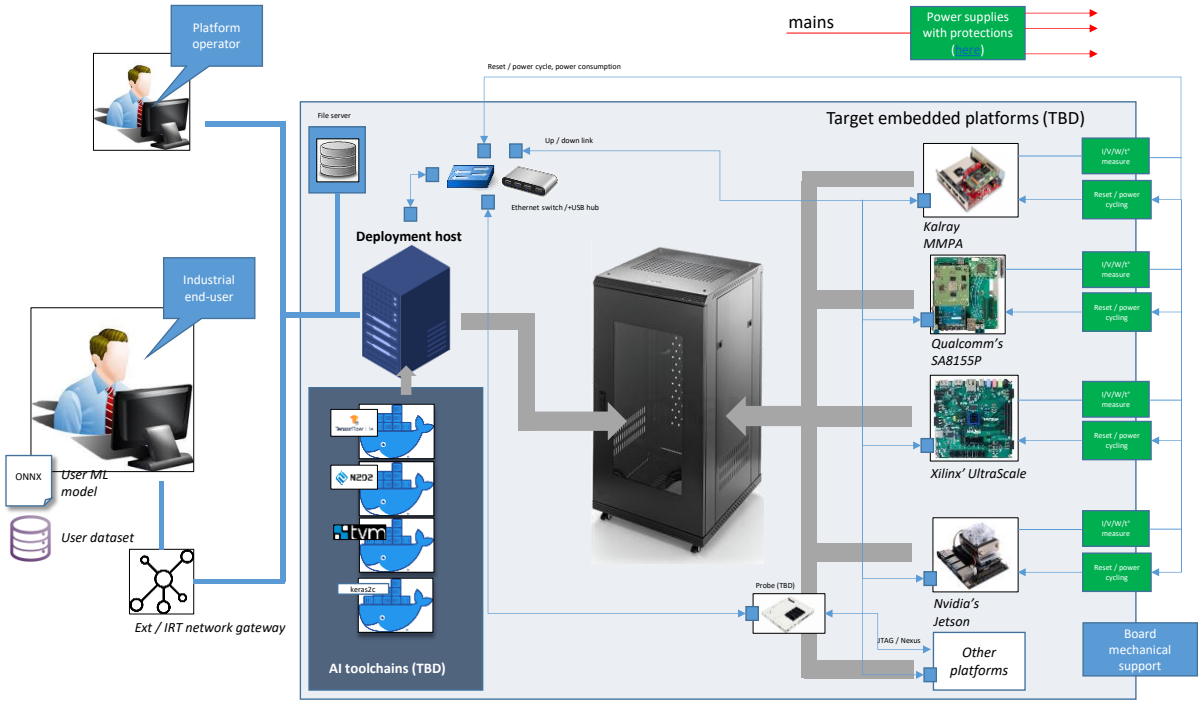


Figure 2. Overview of the benchmarking environment

II – Benchmark EC7.4 definition

A. Target platforms

A.1 Approach

The benchmarking environment provides access to a set of hardware platforms[1] (also called execution targets or target platforms), on which models are deployed, inferences are executed, and performances are measured.

Those platforms have been selected according to several criteria recalled on Figure 3 and briefly recalled hereafter:

- **Feasibility**. In order to ensure the production of useful results in 2022, and considering the current shortage of electronic components, focus has been placed on hardware platforms available and mature in 2022 and with short provisioning delays. This includes in particular platforms already available at our premices. The NG Ultra platform, for instance, will be considered in 2023 since the tool chain still requires to be matured.
- **Efficiency**. Considering the available time and effort available, we have privileged hardware platforms for which the project team has already some experience (e.g., the Zynq Ultrascale) or with a "reasonable" level of complexity (e.g., the raspberry Pi). As a counterexample, the new and complex Xilinx' VERSAL family will be studied and integrated in the benchmark in 2023.
- **Coverage of chip technologies and families.** The benchmarking environment aims at covering a large spectrum of applications and systems imposing specific constraints. Accordingly, our "sample" of hardware platforms account for various criteria including: the nature of the processing units (CPU, GPU, FPGA, TPU,…), the number of those processing units (e.g. mono and many cores), their computation power (e.g., low-end microcontrollers and high-end processor dedicated to High Performance Computing, low and high power consumption GPUs, etc.), their electrical consumption and, also, their family (ARM, RiscV, etc.). This "coverage" is illustrated on Figure 4.
- **Sovereignty.** A particular interest is placed on hardware target developed in Europe or France: Infineon, STMicro, Kalray, GreenWaves, and NXP. Some ones with not sufficient maturity, especially the required deployment tools, should be evaluated next year. The Risc-V family of targets has also been selected for the very same reason.
- **Industrial domains.** Due to historical (and technical reasons (e.g., cost per unit, robustness to radiations, etc.), different industrial domains have made different technological choices. Therefore, in order to comply with the expectation of all users, our selection of targets tries to covers domain-specific targets (e.g., Infineon Aurix for the automotive domain, nanoXplore's NG Ultra for the space),
- **Technological pertinence.** This last criterion can hardly be defined in a formal way… Basically, it aims at capturing technologies that raise a strong interest of the industrial and academic community, *even though* the question of their embarcability if business or safety critical system is not necessarily

Finally, the platforms identified by the industrial end-users in the questionnaire proposed in 2021 have been retained. This is for instance the case of the Qualcom SnapDragon SA8155P used in the Renault/Valeo use case.

[1] In our context, the term "platform" refers to some electronic device able to execute all or part of the inference process.

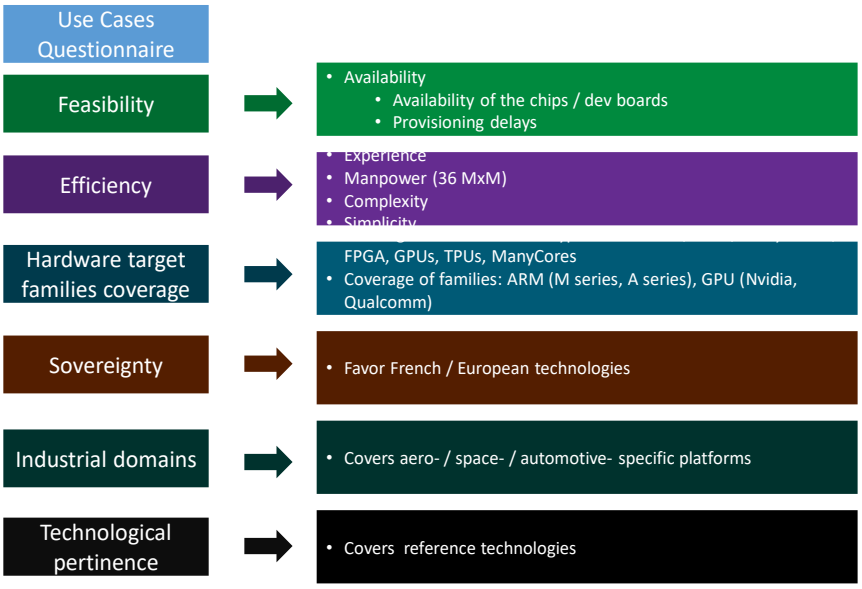


Figure 3. Main selection criteria of the AI models execution target/platform

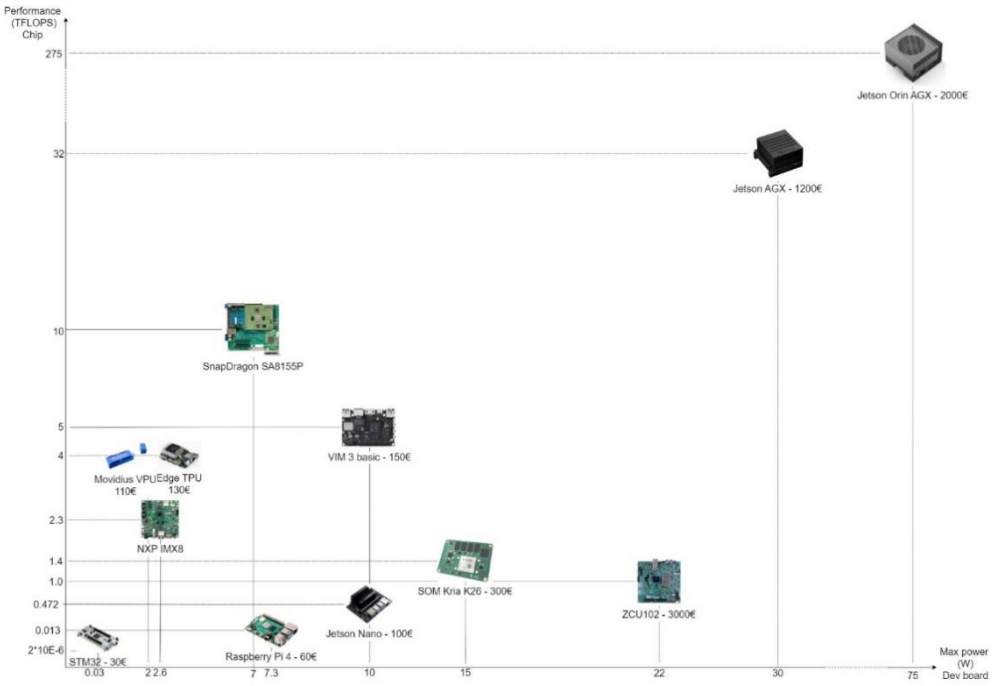


Figure 4. Main performances range of the selected execution targets

At the bottom one MCU STM32 (<1W) to inference tiny ML and at the top the last generation of embeddable GPUs Nvidia Orin AGX (275TFLOPS) to process large AI models, and several intermediate solutions implementing different technologies (RiscV, SoC FPGA, AI specialized ASICs, etc.)

A.2 Attributes

Several attributes have been considered to select the HW platforms. They are listed in Table 1.

Attributes are grouped in 4 categories:

- "GEN": general attributes, mainly useful for managing the EC7.4 benchmarking activity.
- "SOM": attributes of the System On Module, i.e, the development board embedding the electronic chip to be evaluated with the necessary interfaces and additional devices to connect it to the test bench, to deploy the AI model and to perform inferences.
- "SOC": attributes of the System On Chip, i.e., the electronic die implementing the core processing resources used to perform the inferences.
- "+": additional attributes not classified in previous categories.

GEN	Deployment year	Year of deployment of the platform in the benchmarking environment
	Location	Location of the platform on the sites involved in the action E.g., Toulouse, Paris, Sophia
	Responsible	Individual in charge of the platform
	Availability	Availability of the platform for its integration in the environment
	Provisioning delay	Provisioning delay
SOM	Name	Name of the development board hosting the SoC (e.g., ZCU104)
	Price	Price of the development board
	Power consumption (W)	
	Memory (Go)	
	Country of manufacturing	
	Weight (g)	
	Operating Temperature (°C)	
	Interfaces	
SOC	Name	Name of the SoC (e.g., Zynq UltraScale)
	Price	Price of the SoC
	Architecture and frequencies	Data about the SoC architecture (number of cores, architecture of the core, etc.) e.g., "4xA53 [1.2Ghz, SIMD/MIMD] (CPU)+Adreno [X Ghz](GPU)+ Hexagon(DSP)", "ArtixV7 [XXX LUT, YYY BRAM, ZZZ DSP] (FPGA)"
	Manufacturer	Manufacturer of the SoC (e.g., Xilinx)
	Country of manufacturing	
	Power consumption (W)	
	Memory (Go)	Memory embedded in the SoC (on-chip), e.g., 1Gb (flash) + 1Gb (SRAM) + 64Kb (L1 cache) ... BRAM... + bandwidth
	Performances	Data about the SoC raw performances in FLOPS, TOPS, etc. E.g., A53 [100 GFlops, 100 MIPS,...], Adreno [100GLops]
	Grade	
+	OS	OS supported by the board (e.g., Android)
	Supported framework	ML framework supported by the board (e.g., TensorFlow Lite)

Table 1 : Execution platforms attributes

A.3 Selection

Table 2 lists the selected HW platforms to be evaluated on the ML-bench in 2022 and 2023 sorted by order or priority. This order complies with the principles defined in the previous sections. In particular, it covers a large scope of chips technologies, in accordance with the Confiance.ai industrial feedback. The complete table, including all attributes and all boards, is available on the EC7.4 project Sharepoint.

Name	Price	Power cons. (W)	Memory	Name	Architecture and speed	Performance	OS	Supported framework
SnapDragon SA8155P	2900$	7W	LPDDR4 12Go Flash UFS 3.0	SnapDragon SA8155P	SoC ARM V8, 8xKryo 485 +GPU Adreno+DSP Hexagon (A76+A55) 1x Kryo 485 Gold Prime @ 2.84 GHz + 3x Kryo 485 Gold @ 2.42 GHz + 4x Kryo 485 Silver @ 1.80 GHz	CPU: ? GPU: 1TFLops DSP: ? AIP: ?	Android	TensorFlow Lite, SNPE (Qcom proprietary)
ZYNQ Ultrascale + FPGA				ZYNQ Ultrascale + FPGA	SOC FPGA		Linux	VITIS-AI / Pytorch / N2D2 Dneuro
Jetson AGX				Jetson AGX	GPGPU		Linux	N2D2 TensorRT
PI4	100-200	2-6W	LPDDR4 4Gb	Broadcom BCM2711	ARM-Cortex-A72 (4 x 1,50 GHz)	13,5 GFLOPS 240 pts / W	Rasbian 10	ONNX runtime, TF lite, etc
Myriad X	100	NA		Movidius stick NCS2 (USB)	ASIC/TPU		NA	OpenVino
STM32		15-60	564 Kbytes	STM32	MCU (ARM cortex M7)			CubeAI, N2D2 C++ STM32
SOM Kria K26	245	7.5 - 15W		SOM Kria K26	SOC FPGA (ARM cortex-A53), Zynq UltraScale+ MPSoC (XCK26)	XCK26 (FPGA): 256K Logic Cells		VITIS-AI

Name	Price	Power cons. (W)	Memory	Name	Architecture and speed	Performance	OS	Supported framework
Jetson Nano	90-150$	5W	4 GB 64-bit LPDDR4 - Flash : eMMC 5.1 16Gb	Jetson Nano	128-core GPU Max Operating Freqency : 921 Mhz ARM Cortex -A57 MPCore (Quad-Core) Processor with NEON Technology - Max Operating Frequency : 1.43Ghz	472 GFLOPs	Linux	Tensor RT
Coral	99$	3 W	RAM : 2 GB LPDDR3 - Flash : 8 GB eMMC	Coral Accelerator Module	ASIC/TPU Quad-core Cortex-A35 processors: 64-bit Armv8-A architecture - Frequency up to 1.5 GHz	4 TOPS	Mendel Linux	TensorFlow Lite
Nxp IMX8	100-200$	N/A	2GB DDR4 SDRAM 8GB eMMC MicroSD Slot	NXP i.MX 8M	Cortex-A53 (ARMv8-A) + Cortex-M4F (Armv7E-M)	2,3 TOPS	Debian	TensorFlow Lite
VIM3 basic	150	N/A	2 GB LPDDR4	VIM3 basic	Amlogic A311D: x4 Cortex A73 performance-cores (2.2Ghz) + x2 Cortex A53 efficiency-cores (1.8Ghz)	5.0 TOPS NPU	Linux	Tengine SDK
Jetson AGX				Jetson AGX	GPGPU		Android	
Orin AGX	2000-3400	15-60W	64GB eMMC 5.1	Jetson AGX Orin	GPGPU GPU 2048-core NVIDIA with 64 Tensor Cores CPU 12-core NVIDIA Arm® Cortex A78AE v8.2	275 TOPS	Ubuntu 20.04.4	Tensor RT
AURIX 2G TC399				AURIX 2G TC399	µcontroleur			
NG Ultra				NG Ultra	SOC FPGA			
Kalray				Kalray	Manycore			

Name	Price	Power cons. (W)	Memory	Name	Architecture and speed	Performance	OS	Supported framework
Versal				Versal	ACAP			
SiFive Performance P270				SiFive Performance P270	RISC-V			
Greenwave GAP-9				GAP-9		150.8 GOPS		

Table 2 : List of targeted boards in 2022-2023

B. Frameworks

B.1 Approach

In order to be executed on some specific hardware, a set of means including tools, libraries, etc., the model needs to be transformed into a form executable or interpretable by the execution environment. In our context, we refer to this set as the “deployment framework” (or “deployment FW”).

The process of selecting the inference implementation tools is constrained at two levels:

- on the one hand, by the compatibility of these tools with the main ML frameworks used for the definition and training of the selected models;

- on the other hand, by the possibility of running these tools on selected hardware targets, which often require proprietary tools that are compatible with restricted ML frameworks.

These constraints drastically reduce the amount of possible deployment framework for a given task.

The prioritized deployment frameworks are the ones that are specifically dedicated to a certain target.

But because dedicated automatic inference implementation tools may not always be available, multi-steps toolchains (using transcoding libraries like Keras2C) are also considered.

Thanks to the micro-service architecture, one can always integrate new toolchains down the road.

B.2 Attributes

Table 3 lists the attributes characterizing a deployment FW. The most important one is the “execution environment” that determines the compatibility between the model and the HW platforms. It will be of interest to measure the AI metrics applying the quantification proposed versus the same metrics in FP32 to evaluate the “confidence” in the deployment tool.

Attribute	Values	Meaning
Framework	Ex: TensorRT	Identification of the framework
Provider	Ex: Xilinx	Provider of the framework
Open-source	YES	
	NO	
Community	VERY LARGE	
	LARGE	
	SMALL	
	VERY SMALL	

Outputs	C code	The inference implementation chain generate C source code
	bistream (FPGA)	
	.uff	
Execution environment	GPU	
	CPU	
	FPGA	
	Bare metal ...	
Supported optimizations	Quantization	
	Pruning	
Tracability		Capability to trace the generated implementation to the input model
Timing determinism		
Semantic preservation		Capability for the implementation to generate the same results as the input model (in the absence of other optimizations)
Operator Compatibility (ONNX)	<url>	

Table 3 : Attributes of the deployment framework (focus on those relevant to the execution targets).

B.3 Selection

Table 4 gives the list of the frameworks considered in the 2022-2023 period.

Framework	Provider	Open-source	Execution environment	Supported optimizations
TensorFlow Lite	Google	YES	GPU, mobile, microcontrolers	INT8 Quantization, Pruning
TensorFlow for MCU	Google	YES	CPU (Hexagon, Risc-V, STM32), ESP32, TI Dev Boards	Not available
VITIS-AI	Xilinx	YES	FPGA	INT8 Quantization, Pruning
TensorRT	NVIDIA	YES	GPU	INT8 Quantization
OpenVino	Intel	YES	CPU, GPU, GNA, Movidius VPU	Quantization, filter pruning, binarization and sparsity
CubeAI	STMicroelectronics	Yes	STM32	INT8 Quantization
TVM	Apache	YES	GPU, CPU	FP16 Quantization
N2D2	CEA	Partially	MCUs and DSPs (STM32 and ARM cores)	Quantization (INT8, FP16), Quantization Aware Training (QAT)
Pytorch Mobile	Meta	YES	CPU (ARM64), IOS (Metal), Android (Vulkan)	Quantization (INT8), Pruning
Finn	finn.ai		FPGA	Quantization, Quantized Aware Training (with Brevitas)
keras2c	Rory Conlin	YES	Bare metal	Not available

uTVM	Apache	YES	Bare metal,CPU	Quantization
snnTorch	Jason Eshraghian	YES	GPU (with CUDA), CPU (wrapper de Pytorch)	Quantization, Quantization Aware Training (QAT)
BindsNET	BindsNET	YES	GPU (with CUDA), CPU (wrapper de Pytorch)	Not available
Matlab DL HDL Toolbox	MathWorks	NO	FPGA, SoC	INT8 Quantization
Tengine	OPEN AI LABS (KHADAS)	YES	NPU (VIM3 Khadas)	INT8 Quantization

Table 4 : list of deployment frameworks relevant with respect to the selected AI execution targets.

C. Models

C.1 Approach

Within the framework of EC7.4, the choices concerning the Machine Learning architectures must be made in accordance with the **state of the art**, the **use cases of Confiance.ai** and their **scientific interest**. Furthermore, as some layers/operators may not be compatible with certain hardware deployment chains, the availability of these layers will also be a factor to filtering our model choices, which may be downgraded in terms of version.

The families of models that have been shortlisted are described in Figure 5.

Initially, we decided to work on the main families of **computer vision** by Deep Learning, namely: Classification, Object Detection and Segmentation. The case of Reinforcement Learning and Spiking Neural Networks will be considered at a later stage depending on the progress and resources available during the study. These choices are reflected in the choices made by the MLCommons consortium through MLPerf (Figure 6), which also chose to work on Natural Language Processing (NLP) techniques and Recommender Systems.

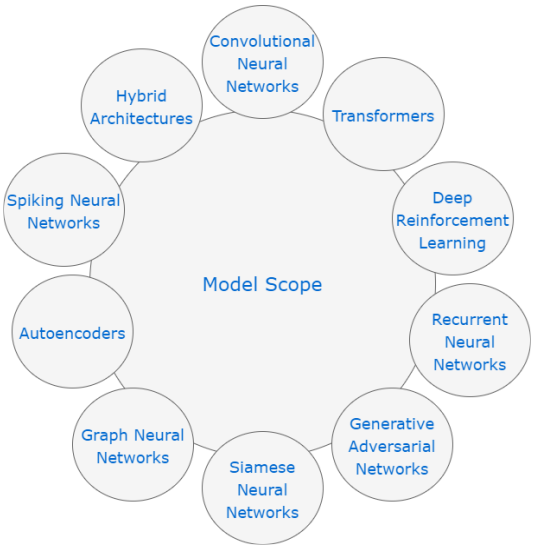


Figure 5. Deep Learning families shortlisted.

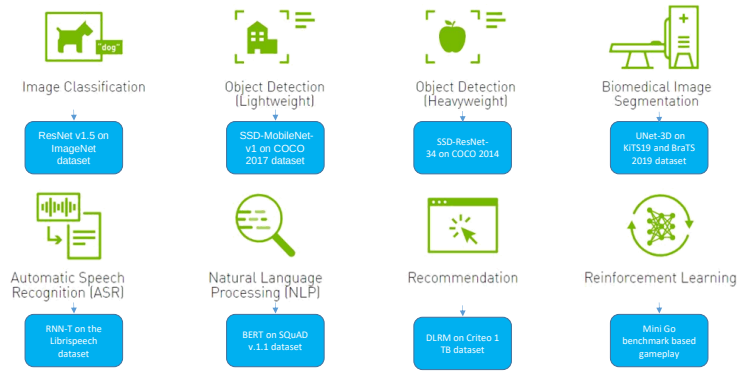
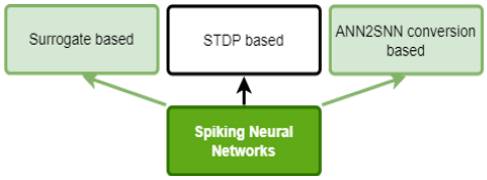


Figure 6 : MLPerf choices [NVIDIA website].

Spiking Neural Networks (SNN) are gaining interest in computer vision (object detection, autonomous navigation, pose estimation, etc.). Natively compatible with event-based sensors, these approaches are promising in terms of memory occupation and latency, and can improve the solution proposed for the embeddability of neural network-based learning models. The embeddability of these networks remains a challenge today because few open-source deployment chains support them, thus implying a significant development overhead (e.g. in VHDL for FPGA deployment).

Selected spiking model types

- Surrogate based : supervised spiking learning
- STDP based : semi unsupervised spiking learning
- ANN2SNN conversion based : supervised "formal" learning



Implementation methodology

- The most known inference implementation tools (CubeAI, TF Lite, Vitis AI, …) do not support SNNs
- Code "inference program" for each HW target
  - C/C++ for CPU/MCU
  - VHDL/HLS for FPGA
  - Python (CUDA) for GPUs
  - Others ?

Figure 7: Highlight on the specificities of the Spiking Neural Networks

C.2 Attributes

Some attributes listed in the Table 5 are relevant to select the git/trained. For instance a routine was developed to analyze the compliance of the DL layers with the deployment tools applicable to the selected execution targets.

Attribute	Values	Description
Priority		The first models to be deployed
Family	CNN (Convolutional Neural Networks)	
	RL(Reinforcement Learning)	
	RNN (Recurrent Neural Networks)	
	GAN (Generative Adversarial networks),	
	SNN (Spiking Neural Networks)	
	GNN (Graph Neural Networks)	
	AE (Autoencoders)	
	SOM (Self Organizing Map)	
	CAP (Capsule Neural Network)	
	HA (Hybrid Architectures)	
Model		Actual model (e.g., ResNet, YoLo, etc.)
Application		Typical usage of the model (e.g., image segmentation, NLP, etc.)
Number of parameters		Number of parameters (typical weights/biases) of the model
Link		Link to the documentation
Performance		Typical metric according to the test set
Framework		Coding and training libraries
Available format	(.pt etc.)	Reflecting the FW used
Dataset		Dataset used to retrieve the weights and computing performances
Source code	Link to the code	

Table 5 : attributes relevant to AI models selected

C.3 Selection

This initial selection has enabled us to highlight a large number of state-of-the-art models in a wide variety of applications. However, given the manpower available on this work package, it was necessary to refine this selection. For this, we based ourselves on the selection (at the family level) that was made on the MLPerf project (originating from the MLCommons consortium) as well as on the needs of the Confiance.ai project partners. In addition, we have chosen to produce, for each selected family, a model with latency constraints (lightweight) and a model with performance constraints (heavyweight). This logic is also reflected in our hardware choices, where we have selected,

for each target family (GPU, NPU, FPGA, etc.), a low and a high-power card (considering TFLOPs and W). In the selection criteria, we have also considered the availability of the model trained on a state of the art dataset (Imagenet, COCO, etc. Depending on the task).

The resulting selection is summarized in Table 6.

Family	Model	Application	Nbre param	Performance	Framework	Dataset
CNN	MobileNetV2	Image classification	3.4M	73% (Top-1 Acc)	Pytorch	ImageNet
CNN	ResNet50 + Yolo	Object Detection	**N/D**	**N/D**	ONNX	Woodscape
CNN	MobileNetV3s	Image classification	2.5M	67.4% (Top-1 Acc)	Keras, Pytorch, TF	ImageNet
CNN	YOLOv7	Object Detection	36.9M	51.2% (AP)	Pytorch	MS COCO
CNN	MaxViT-B	Image classification	120M	86.66% (Top-1 Acc)	Tensorflow	ImageNet
CNN	UNet	Image Segmentation	8M	92.0% (MIoU)	Pytorch	ISBI
CNN	EfficientNetB3	Image classification	12M	81.6% (Top-1 Acc)	TensorRT script available	ImageNet
CNN	DeepLabV3+	Image Segmentation	59.5M	89.0% (MIoU)	Pytorch	Cityscapes, Pascal VOC
CNN	PIDNet-S	Semantic Segmentation	7.6M	78.8% (MIoU)		Cityscapes
CNN	DETR	Object Detection	60M	44.9% (AP)	Pytorch	COCO
CNN	RetinaNet	Object Detection	57M	40.8% (AP)	Pytorch	COCO
RL	Deep Q-Learning (DQN)	N/A			Pytorch	Gym
RL	Proximal Policy Optimization (PPO)	N/A			Pytorch	Gym
RL	Synchronous Advantage Actor Critic (A2C)	N/A			Pytorch	Gym
SNN	Surrogate based : LeNet	N/A				MNIST (ou CIFAR-10?)
SNN	ANN2SNN : LeNet	N/A				MNIST
SNN	STDP based : LeNet	N/A				MNIST

Table 6 : this table lists the AI models to be deployed on the selected HW platforms by order of highest priority first

Confiance ai

Titre : Definition of target hardware platforms, frameworks and models

Mots clefs : Machine Learning, évaluation de performance

Ce document identifie les plateformes matérielles, les environnements d'implémentation (*frameworks)* et les modèles qui seront intégrés dans l'environnement d'évaluation de Machine Learning (ML *bench*). Il établit en outre la liste des attributs caractéristiques de chacun de ces éléments.

Title : Definition of target hardware platforms, frameworks and models

Keywords : Machine learning, benchmarking

This document identifies the hardware platforms, the frameworks and the models that will be integrated in the Machine Learning Benchmarking Environment (ML Bench). It also provide the list of attributes used to characterized those elements.