EC7

ML Benchmarking Environment Specification

Specification of the Confiance.ai benchmarking environment

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 F. FERESIN O. HLIMI E. JENN Ch. MARABOTTO F. THIANT Ph. THIERION	IRT Saint Exupery ATOS AzurIA IRT Saint Exupery IRT Saint Exupery IRT Saint Exupery IRT SystemX Renault	Research engineers

Document control

Revision	Date	Commentary	Author
1.0	2022/07/05	Creation	See list of co-authors above.

Contents

I - Introduction

A. Introduction

This document gives

- the operational needs for a *benchmarking environment* (Section B)
- the requirements for the *benchmarking environment* (Section C).

Operational needs

A. Operational needs

A.1 Introduction

This chapter defines the operational needs for a benchmarking environment.

We distinguish two categories of users:

- *The industrial **end-users***
- *The **operator** in charge of maintaining the environment*

A.2 Operational needs

A.2.1 Definitions

The following terms are "factored out" in order to simplify the expression of the operational needs.

- <application> is <algorithm> | <model> | <model><data>
  - *This identification of "application" is aimed at allowing the benchmarking environment to be used on problems with different levels of maturity, from the case where the learnt model and the dataset are available up to the point where the user only knows the "type of algorithms" that he/she plans to use.*

- <algorithm> is *(list TBD)*
  - *The objective is to provide a first (and rough) level of discrimination between various high-level categories of algorithms such as "object detection", "image segmentation", "NLP", etc.*
    At that level, this is more a question of documentation (e.g., task / network type matrix) that should be collected in EC3,4 and in the literature…

- <model> is <open model> | <user provided model>
  - <open model> is (list TBD)
  - *An <open model> can be embedded in the benchmarking environment whereas a <user provided model> shall be uploaded.*

- <dataset> is <open data> | <user provided data>
  - <open data> is (list TBD)
  - *An <open data> can be embedded in the benchmarking environment whereas a <user provided model> shall be uploaded.*
  - *If the data are provided by the user, it means that some pre-processing may be needed. In addition, the data may be embedded in a flow (e.g., a video) that needs to be processed. The pre-processing being application-dependent, this means that we have to provide means for the user to describe its*

preprocessing. This could be done (e.g.) in Python ; but what about the embedded targets? In that case, the code is tightly dependent on the target (and on its attached devices).
We have to clarify the "scope" of the tool: the objective is not to provide a generic purpose development environment…

- <target environment> is <deployment scheme> x <hardware target> x <execution environment>
  - <deployment scheme> is <deployment chain> <deployment parameters>*
    - <deployment chain> is *(list TBD)*
      - *See the excel sheet.*
    - <deployment parameters> is *(list TBD)*, includes quantification,…
  - <hardware target> is <existing hardware target> | <non existing hardware target>
    - <existing hardware target> is *(list TBD)*
      - *See the excel sheet.*
    - <non existing hardware target> is *a platform "similar to some existing platform" but not available.*
      - *The objective is to provide a means to "extrapolate" metrics from those already available. See work from Smail et al.*
  - <execution environment> is *(list TBD)*
    - *The actual list depends on the target. Examples: bare metal, PetaLinux, VxWorks, etc.*
- <optimization problem> is <application> <cost function> <constraint>
- <cost function> *involves* <metric>+
  - *A cost function involves one or several metrics. The cost function needs to be minimized.*
  - <metric> is <implementation metric> | <functional metric>
    - <implementation metric> is "memory usage" | "latency" | "throughput" | , "energy consumption" | "temperature"
    - <functional metric> is <accuracy metric>
    - <accuracy metric> is "accuracy" | "precision" | "recall" | "F-score" | "IoU"| …
- <constraint> is <metric constraint>* <other constraint>*
  - <metric constraint> is *a constraint involving a* <metric> (e.g., "max inference latency < 100ms", "mean latency < 100ms", "standard deviation of latency < 0.9", etc.
  - <other constraint> is <licence constraint> | <openness constraint> | …

A.2.2 Operational needs for the end users

1) An easy to use interface (IHM with scrolling menu?) shall be provided with clear access to several options at minimum:
   a) the evaluation tool to perform a preliminary selection of the compliant <target environment>
   b) the "benchmarking tool" to perform a model deployment on selected target(s)
2) The user shall have the possibility to select several high-level criteria:
   a) the HW attributes (as defined in "HW platform attributes" in table "EC4_Charge-criteres.xls") or the required <existing hardware target> if available
   b) *the AI model* if available in <open model> with associated <open data> or its own model and associated test dataset (option to be confirmed)
3) The user needs to check if his / her <application> can be deployed on a given <target environment>.
4) The user needs to select the best <target environment> for his / her <application> according to some <cost function> and a set of <constraints>
5) The user needs guidance to select a <target environment> for a given <context> and set of <constraints>.
   a) Identification of valid framework + hardware combination for the problem at hand
6) The user needs to evaluate some <metric> for his/her <application> with a certain <confidence level>.
   a) Refer to "Metrics" in table of EC4_Charge-criteres.xls
7) The user needs to reproduce the deployment chain used in the experiment in his/her own site.
   a) The system configuration shall be documented and the setup shall be easily reproducible.
8) The user needs to perform series of experiments with the minimal effort.
   a) The system shall provide means to automate a series of experiment without user monitoring and control.
9) The user needs to retrieve the results of experiments in order to analyse and present them.
10) The user needs to reuse existing experiment results without re-executing those experiments.
    a) The results must be archived and versioned.
11) The user needs to be informed, for instance by a "popup", of the emergence of new <deployment chain>, <hardware platforms> (technological watch)
12) The user needs to justify the choice of a <deployment scheme>, <hardware target>, <execution environment>. The results of the evaluation must be completed by justification elements about the deployment chain and hardware platform
    a) The system shall provide document about the features of the deployment tool and target platform in order to justify (to some extent…) the the selected <target environment>.
13) The system shall guarantee the security (confidentiality, integrity and availability) of data (incl. models, datasets, toolchain parameters, results, etc.).

A.2.3 Operational needs for platform operator

1) The platform operator needs to reproduce the complete benchmarking environment on another machine in case of a failure of the benchmarking platform.
2) The platform operator needs to add new <target environment> to the environment.
3) The platform operator needs to diagnose the case where the deployment or execution has failed in order to fix it.
   a) The user needs to be informed that the deployment or the execution has failed
   b) The platform operator needs to have appropriate data to diagnose the problem.
4) The system needs to be fitted with a mechanism to log data about the deployment and execution process.

A.2.4 Operational constraints

1) The service shall be remotely accessible.
   a) The service shall only be accessible to registered users.
   b) The model / dataset provided by the user shall be secured (the security policy is to be formally defined).
2) Execution of a end-user experiment shall require no action from the platform operator (e.g., no installation, no reset, power-cycling, etc.).
   a) The experiments (deployment, execution, measurement, reinitialization) shall be fully automated. In particular, it shall require no manual operation (to load the code onto the target platform, to power cycle / reset the platforms, etc.).
   b) Should the platform fail, recovery shall be automatic (watchdog,…)
3) The user (end user and platform operator) shall be informed of the state of execution of an (a series of) experiment(s)
   a) The system shall provide a monitoring panel.
4) The result of an experiment shall not depend on previous experiments or on experiment performed concurrently on other HW platformsThe system shall isolate experiments. In particular, a target platform shall be placed in a well-known state before each experiment
   a) The system shall reset a target before each experiment. ("experiment" to be defined)
   b) The system shall isolate experiments. In particular, a target platform shall be placed in a well-known state before each experiment
5) Computational resources shall be shared evenly among users (no user shall be able to monopolize the platform). A resource usage policy must be defined and enforced.
6) Executions of experiment shall exploit all available hardware targets compatible with the experiment definition.
7) When several experiments requires the same target, they shall be executed in FIFO order (in compliance with the resource usage policy).
8) The hardware resource usage shall be logged on a per user and per experiment basis.
   a) Adding a new <target environment> shall have no impact on the existing <target environment> (in particular, there shall be no need to modify an existing tool chain when adding a new tool chain).
   b) In case of a HW failure, the unavailability of the platform shall not exceed the time of provisioning of the hardware + 1 day.

System requirements

B. System requirements

B.1 Introduction

This Chapter defines the system requirements of the benchmarking environment.

B.2 Definitions

- <performance metrics>: <ML performance>, <semantic preservation>, <memory usage>, <inference time>, <power consumption>, <energy consumption>. When applicable, measures shall include min, max, mean, and standard deviation.
    - Comment: other metrics such as temperature, cache hit/miss, bus throughput, number of instructions executed per inference may also be considered depending on the target platform.
    - Comment: other AI-related metrics could also be considered, such as *robustness.* To be checked with EC3 and EC4. In a future version of the benchmarking environment, we could provide tools to generate adversarial examples and check the relative robustness of the deployed model. As an intermediate step, the user may provide a robustness-evaluation dedicated dataset , and simply evaluate the response of the different implementations).
- <memory usage>: <flash memory usage>, <static data area usage>, <stack usage>
    - Comment: the type of available memories depend on the platform. For a TC3XX, for instance, one has to consider the local DSPR, PSPR, local LMUs, remote LMUs, etc. For a given platform, the environment shall return a list of memories with their relative usage.
- <semantic preservation>: for the same set of inputs, <sematic preservation> is measure as the ratio of <identical outputs> between the <input model> executed in the training environment and the <deployed model> executed on the target platform. This metrics is aimed at showing to what extent the deployed model produces the same result as the training model.
- <identical outputs>: two outputs are considered identical if their distance (according to some distance metric) is under a (user-defined) threshold.
- <input model>: the model deployed in the training environment .
- <deployed model>: the model deployed in the target environment
- <experiment>: estimation of one or several <performance metrics> for a given <experiment configuration>.
- <experiment configuration>: tuple (<inference problem>, <inference implementation definition>, <execution target>, <execution environment definition>)
- <execution target>: actual hardware target (i.e., a board, a desktop computer, etc.), simulated model of a hardware target (e.g., a simulator, a virtual platform), or descriptive model of a hardware target (i.e., a

model giving the main attribute of the platform such as type and number of cores, clock frequency, memory size, etc.).

- <execution environment definition>: couple (<inference execution environment>, <inference execution environment configuration>).
- <inference problem>: tuple (model, input dataset, [expected outputs]). The expected output may or may not be provided depending on the need to estimate <accuracy> and the <semantic preservation>.
- <inference implementation definition>: couple (<inference implementation tool>, <inference implementation configuration>).
- <inference implementation configuration> is the set of values determining the behaviour of the inference implementation tool. This includes, for instance, the type of optimization applied, the quantization applied, etc. The set of parameters depend on the tool.
- <fair usage> = usage compliant with the resource usage policy. For instance: no resource can be used more than X% of the time in a 24h period.
- <optimization domain> is a set of <constraints> that the solution (i.e., <inference implementation definition>, <execution target>, <execution environment definition>) must comply with to be acceptable.
- <optimization criterion> (or "cost function") is a function involving one or several <performance metrics> that need to be minimized by the optimization process.
- <optimization strategy> describes how the solution space (<inference implementation definition>, <execution target>, <execution environment definition>) must be explored.
- <guidance> is any information, hint, fact, that can help the user to select an appropriate tool, target supporting his/her problem. Guidance may take various form such as: "This platform cannot support the provided model because it does not provide sufficient RAM for the model", "This tool cannot support the provided model because it does not support operator XXXX."
- <deployed model> = <inference model>: the model to be executed on the target platform.
- <training model> = <input model> model before deployment.
- <dataset>
- <ML performance metrics>: <ML performance metrics for classification and detection>, <ML performance metrics for regression>
- <ML performance metrics for classification and detection> : <precision>, <recall>, <F1-score>, <Intersection over Union (IoU)>
- <ML performance metrics for regression>: <Mean Square Error (MSE)>, <Mean Absolute Error (MAE)>, <Root Mean Square Error (RMSE)>, <R2 Coefficient of Determination>, <Adjusted R2>

B.3 Specification

B.3.1 Benchmarking requirements

The benchmarking of hardware platforms requires to follow a benchmarking procedure, allowing the measurement of ML metrics (e.g. accuracy) and hardware metrics (e.g. : latency, power consumption). In case of embedded devices, the collected ML measures should be retrieved from the targets, via a communication canal (e.g. : TCP, Serial), and integrated into the tool managing the ML lifecycle (e.g. : MLFlow). The collected hardware measures are the most often accessible via dedicated hardware probes and low-level API, strictly hardware dependent.

Furthermore, for the sake of transparency and reproductibility, the experiment environment leading to these measures should be packaged and easily deployable on a host machine (e.g : docker).

REQ- 00001: Estimation of performance metrics

The system shall allow the estimation of <performance metrics> for a set of user-defined <experiment configurations>.

■

REQ- 00002: Provision of problems

In the <experiment configuration>, the <inference problem> may be either provided by the user or selected by the user among a predefined set of benchmarking models.

■

REQ-00003: Independence of experiments

The system shall ensure the independence of each experiment with respect to other experiments performed using the system. In this context, independence means that the results of an experiment shall not depend on another experiment executed before or concurrently.

Furthermore, to provide an optimal environment of benchmarking, with the less disturbances as possible, only one single inference should be running on an embedded device, this one reinitialized in a clean and controlled state between runs (i.e. cleaning of registers etc.)

■

REQ-00004: Reproductibility

The system shall ensure the reproducibility of experiments. This means that all conditions (about the tool chain, the execution platform, the measurement devices, etc.) that may have an impact on measurements must be identified, possibly observed and ideally controlled. For conditions that are controllable, the system shall have the capability to set them before executing an experiment. For conditions that are observable, those conditions must be part of the experiment results provided to the end user.

■

REQ-00005: Storage of input and output data

The system shall store each experiment input and output data. The format of the data must be defined and documented.

- Rationale: the objective is to support a later consultation and retrieval of results by the end user

■

REQ-00006: Consultation of past experiments

The system shall allow the user to consult, download, and display previous <experiment configurations> and <experiment results>.

■

REQ-00007: Saving of experiment data

The system shall allow a user to save or discard the results at the end of an experiment.

■

REQ-00008: Management of previous experiment data

The system shall allow a user to suppress data concerning previous experiments.

■

REQ-00009: Replay of experiments

The system shall allow the replay of an experiment.

> Comment: this means that all data needed to replay an experiment shall be stored. This covers in particular all input data, all configuration data, the version of all tools involved in the workflow, etc. Those data should be part of the experiment definition. For instance, as part of the experiment definition, the user will select one or several toolchains (including versions of frameworks, services, etc.) in a list of predefined toolchains (those available in the environment).
>
> Comment: We do not require that the re-execution of an experiment returns the same results.
>
> Comment: The system may store intermediate results in order to accelerate

■

REQ-00010: Tracability of experiment results

The system shall ensure the traceability of results to the <experiment configuration> used to produce those results and to the user that has executed the experiment.

■

REQ-00011:

The system shall provide information about the execution of the experiment workflow (e.g. logs) to give confidence on the correct execution of the experiment, on the correct data.

> Comment: the objective is not to *prove* that the experiment has been correctly executed, but to give a "minimum level of confidence" about it.

■

REQ-00012: Resource reservation

The system shall support the reservation of a target platform during a well-defined time interval. The reservation shall be compliant with the resource usage policy.

- Rationale: during the development / debugging of the inference application, a "batch" execution model is not appropriate: therefore, a user must be able to reserve the resource during a well-defined time interval.

■

REQ-00013: Fair usage of target boards

The system shall ensure a <fair usage> of the target boards. A platform usage policy shall be defined. The policy shall not prevent a resource to be used it is free.

- Rationale: the hardware resources being limited, they have to be shared between users.

Comment: One possibility could be to implement a replenishment policy in order to ensure a maximum mean level of usage. Additionally,

■

REQ-00014: Series of experiments

The system shall support the execution of a series of <experiments> for a given user.

■

REQ-00015: Monitoring of experiment execution

The system shall provide the end-user with the capability to monitor the execution of a series of experiments.

■

REQ-00016: Control of experiment execution

The system shall provide the end-user with the capability to abort / suspend / resume an experiment or a series of experiments.

■

REQ-00017: No operator action

The system shall execute the series of experiments autonomously (with no operator action).

■

REQ-00018: Signaling of experiment completion

The system shall inform the end-user of the completion or the failure of the submitted series of experiments (e.g., send a mail when the experiment is completed).

■

B.3.1.1 Display and exploitation of experiment results

REQ-00019: Exportation of results

The system shall allow exporting the experiment results

- Rationale: the experiment results may be processed by the end user.

REQ-00020: Presentation of results

The system shall allow presenting (possibly graphically) the experiment results

REQ-00021: Comparison of results

The system shall allow comparing results produced by different experiments.

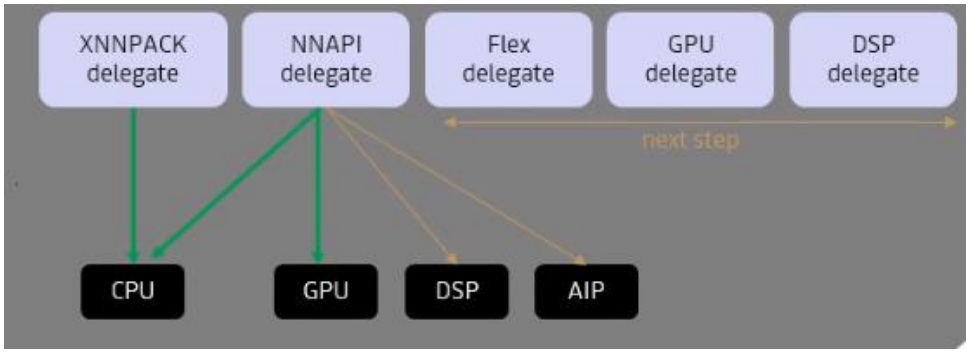
B.3.1.2 Inference code setup

REQ-00022: Configuration of the deployment workflow

The system shall support the fine grain configuration of the deployment workflow.

- Rationale: there might be several workflows (or paths) involving the same toolchain, in particular when model conversion are involved. In that case, the user shall be able to configure precisely the sequence of operations (including conversions) to be used.

  For instance, there may be several possible option, see example below for TensorFlow Lite.



A meta-model of the tools and boards must be defined so as to allow a complete configuration.

REQ-00023: Configuration of tools.

The system shall support the complete configuration of the tools involved in the toolchain (including: quantification level,…).

Comment: identification of the configuration parameters is done in the toolchain workflow diagrams.

■

REQ-00024: Checking of model / toolchain compatibility

The system shall provide information about incompatibilities between the input format and the toolchain.

Comment: This includes "complete incompatibilities" (format not supported) or partial incompatibilities (e.g. Conversions introducing matrix transposition leading to strong performance degradation).

■

REQ-00025:

The system shall provide information about incompatibilities between the input model and the toolchain.

- Rationale: the objective is to prevent or detect incompatibilities as soon as possible (in particular, before performing the experiment).

! Nota: we shall defined precisely the meta-model of an experiment. It shall include the description of the input data, the metrics to be measured, the cost-function to be evaluated, the optimization to be applied, the configuration of the toolchain, the configuration of the target, etc. This meta-model will be used to build the configuration models (or configuration files) that will be used by the tools. This must be defined once for all (but possibly updated…) in order to start building consistent toolchains.

B.3.1.3 Board setup

REQ-00026: Execution environment configuration

The system shall support the precise configuration of the execution environment, including: power cap, running stack (Android), etc.

- Rationale: For some environment (e.g., Android), some part of the processes running in the environment may be stopped by a simple command ("apb stop" for Android). This represents a "second" way to setup an environment with respect to the one consisting to build a specific -- possible stripped down -- environment. See specific requirement for power capping.

B.3.1.4 Performance measurement

REQ-00027: Performance measurement

The system shall provide means to measure performance for one inference of one batch.

Comment: measurement of series of inferences involves specific features/capabilities that are hardware/function-specific (e.g. Image streaming via a camera) and can hardly be implemented for all use cases. This could be investigated in a second phase.

-

REQ-00028: Measurement conditions

The system shall ensure that measurements are performed in known and mastered conditions.

- Rationale: in some conditions e.g. related to hardware initialization, the first inference may take orders of magnitude longer than subsequent inferences. This effect must be identified, possibly avoided and at least documented.

-

REQ-00029: Filtering of measurement noise

The system shall allow *a priori* or *a posteriori* filtering of the measurement noise due the environment (for all measurements). This include, for instance, the effects of other tasks or interrupt drivers being executing concurrently with the inference code.

- Rationale: for complex execution environments, some measures (e.g., latency) may be perturbed by concurrent activities. Those perturbing activities shall be kept to a minimum thanks to an appropriate configuration of the execution environment (e.g., bare metal, simple RTOS, simple Linux configuration, etc.). When all perturbation cannot be removed, the environment shall provide means to filter out these effects. This can be done, for instance, by replicating the same inference multiple times in order to minimize the effect of a perturbation. Another possible solution consists to measure the variations of execution times for a reference workload (e.g., for periods 10ms, 100ms, 1s etc.) and to apply a correction to subsequent measurements.

-

REQ-00030: Semantic preservation

The system shall support the estimation of the <semantic preservation> between the training and the inference model.

Comment: by semantic preservation, we mean the "correspondence" between the results produced by the training mode and the results produced by the deployment model. *To what extent do both models produce identical results?*

- Rationale: In order for the measurement to be meaningful, one has to know whether deployed model behaves "according" to the training model or not.

-

REQ-00031: Accuracy estimation

The system shall support the estimation of the deployed model accuracy.

- Rationale: In order for the measurement to be meaningful, one has to know whether deployed model behaves "according" to the training model or not.

■

REQ-00032: Resource usage estimation

The system shall support the estimation of the memory usage (flash, RAM, stack).

■

REQ-00033: Inference latency estimation

The system shall support the estimation of the inference latency.

■

REQ-00034: Implementation efficiency

The system shall support the estimation of the number of instructions.

- Rationale: this allows the estimation of (i) the "efficiency" of the inference toolchain (in terms of number of instructions required to perform an inference, (ii) the estimation of the IPC.

Comment: this will only be implemented on boards with appropriate performance counters.

■

REQ-00035: Memory throughput

The system shall support the estimation of the memory read/write throughput.

■

REQ-00036: Temperature evaluation

The system shall support the estimation of the chip temperature (100ms sampling rate). Note that for this measure to be independent from previous experiment, some delay may be necessary between experiments.

- Rationale: temperature is an indirect measurement of power consumption. It is not accurate but is easy to obtains (chips are often fitted with temperature measurement).

■

REQ-00037: Electrical power consumption

The system shall support the capture of the electrical power consumption profile during inference (20ms sampling rate).

■

REQ-00038: Performance control

The system shall allow the control of performances (e.g., using DVFS) for platforms supporting this feature.

- Rationale: this feature would allow estimating the performance of a stripped down version of a high-end GPU. For instance, estimating the latency of an 80W 3080 GPU using a 300W 3080 GPU. On an Nvidia GPU, this could be achieved using nvidia-smi command.

■

REQ-00039: Cache usage

The system shall support the estimation of cache hits/misses.

- Rationale: this information gives an indication of the problem / platform adequacy with respect to memory

■

REQ-00040: Initial conditions

The system shall support the precise definition of the inference initial conditions. This includes, for instance, the capability to execute an inference with input data already in GPU memory.

- Rationale: the idea is to support estimations in streaming conditions where the data would be placed directly in memory (e.g. GPU memory) without being first copied from the CPU memory to the GPU memory (using e.g. RDMA).

■

B.3.1.5 Performance extrapolation

(This part will be completed when the work on predictive model starts.)

REQ-00041: Inferred performance estimation

The system shall support the estimation of <performance metrics> for non-available <execution targets>. Extrapolation can be achieved using analytical models or machine learning models.

- Rationale: the objective is to provide the capability to estimate inference performance for an execution platform not available.

■

B.3.1.6 Deployment

REQ-00042: Automatic deployment

The system shall support the setup and deployment of the target execution environment according to the user parameters. Deployment of the execution environment on the target shall not require human action (e.g., connecting a probe, inserting a SDCARD, etc.). Whenever possible, using the TFTP protocol shall be privileged.

■

B.3.1.7 Guidance and optimization

Nota: the following requirements concern a specific, separated, component of the benchmarking environment.

■

REQ-00043: Guidance

The system shall provide <guidance> to select a (toolchain, target platform) combination for a given input model. Ideally, the system should determine the <most appropriate> couple (toolchain, target platform) from the definition of the problem (a model or some abstract description of the model in terms of type of operators, number of layer, type of network, etc.) and set of constraints, without relying on the deployment of the actual model.

■

REQ-00044: User-defined exploiration

The system shall support the user-defined (or "manual") search for the optimal configuration, for a user-defined <optimisation domain>, <optimisation criterion>, <optimization strategy>.

- Rationale: in a first development phase, the benchmarking environment shall support the evaluation of the performance metrics for a given experiment. Optimisation is performed by building a configuration (eg. using some API), realizing the experiments, modifying the experiment (e.g. toolchain configuration, platform configuration, etc.) and looping until some condition is satisfied.

■

REQ-0420 : Optimization

The system shall support the automatic search for an optimal configuration for a user-defined <optimisation domain>, <optimisation criterion> and <optimisation strategy>.

■

B.3.1.8 Documentation

REQ-00045: Documentation

The system shall provide access to the documentation of the models, tools, and boards involved in the experimental setup. This documentation shall include in particular:

- how to setup the platform / tool
- the characteristics of the platform / tool pertinent with respect to ML deployment.

■

REQ-00046: HW platform feature analysis

The system shall provide information about the HW architectural features having an impact (positive or not) on the performance.

- Rationale: we shall not only provide the technical capability to execute a model (ie. setup the toolchains), but also provide information about the reason why such or such result is obtained or is expected to be obtained. This is an essential added value of our work. This work will be done in a later phase of the environment development, when first analyses are completed.

■

REQ-00047: Replicability

The system shall support the reproduction by the end-user of the setup corresponding to the <inference implementation definition> and <execution environment definition>.

B.3.2 Quality of service

B.3.2.1 Response times

REQ-00048: Provision of response times

Response time requirements are intrinsically bounded to the hardware characteristics of embedded targets. Based on empirical experience, i.e. previous experiments performed on the testbench, an average time and delay should be provided to the user allowing it to schedule it experiments.

Likewise, the user should be informed of the scheduling policy running on the testbench, and it impact on the execution times of it experiments.

■

B.3.2.2 Availability

REQ-00049: Unavailability

The system shall ensure a maximum unavailability of 1 day in case of a complete loss of the hard disks.

> Comment: This implies in particular the capability to restore the toolchains in an quasi-automatic way and the capability to archive/restore data.

■

REQ-00050: Retries

The system shall retry a failed action for transient failure conditions. The number of retries shall be bounded.

■

REQ-00051: Hardware failure signaling

The system shall inform the operator in case of a hardware failure.

> Comment: The type of failures have to be defined. We have to keep things as simple as possible. No need to create complex autotest for each board. Maybe a simple heartbeat…

■

REQ-00052: Crash tolerance

The system shall tolerate a crash (OS crash, power failure, etc.). In that case, it shall be able to resume the interrupted experiments (in a consistent way).

■

REQ-00053: Hard disk failure tolerance

The system shall tolerate a single hard disk failure in a transparent way (no data loss, no interruption of service).

■

B.3.2.3 Maintenance

REQ-00054: Operator view of operations

The system shall provide the operator with a global view of the experiments queue, and on-going experiments.

■

REQ-00055: Operator board control

The system shall provide a means for the operator to disable / enable a given board (ie. make it unavailable for a certain duration).

■

REQ-00056: Operator operation control

The system shall provide the capability for the operator to manage (start/stop/resume/abort) experiments.

■

REQ-00057: Communication with users

The system shall provide means for the operator to communicate with the end users.

- Rationale: the end users must be informed in case of failure / maintenance operation / etc.

■

B.3.2.4 Capability of evolution

REQ-00058: Introduction of new tools

The system shall support the introduction of a new deployment tool with a minimal impact on the rest of the benchmarking environment.

■

REQ-00059: Introduction of new hardware targets

The system shall support the introduction of a new hardware target with a minimal impact on the rest of the benchmarking environment.

■

REQ-00060: Multiple version of tool

The system shall support the cohabitation of multiple versions of the same tools (e.g., version with different dependencies).

■

B.3.2.5 Supervision requirements

Firstly, the availability of hardware targets, i.e. the number of running process, next scheduling priority and index, and their status (e.g. : hardware failure, idle etc.) should be provided to the user.

Secondly, the whole pipeline process (data/model uploaded/selected, deployment etc.) should be monitored in real-time, and report any failure in the process to the user.

And finally, all ML and hardware metrics collected during experiments should be integrated into this global monitoring system.

REQ-00061: Provision of diagnosis data

The system shall provide the end-user (resp. platform operator) with the necessary data to support the diagnosis and correction of errors leading to the failure of an experiment (compilation error, conversion error, hardware failure, etc.).

■

REQ-00062: Logging of activities

The system shall log all actions (code generation, cross-compilation, etc.) performed during an experiment. (The unit of logging must be defined.)

■

B.3.2.6 Ergonomic constraints

REQ-00063: Control using Python

The system shall be controllable via a Python script (from the creation of the experiment to the control of its execution, to the display of the results)

- Rationale: This allow the most flexible usage of the tool. In particular, this will allow a user-implemented optimisation process (before the environment integrates this capability).

! This capability is essential. It requires a well-defined definition of the experiment definition meta-model (meta model for the input definition, tool configuration, target platform configuration, etc.).

■

B.3.2.7 Security requirements

As of end of 2022, this section is very preliminary since focus is placed on functional features. Currently, security relies on standard IT practices.

Standard security requirements:

- identification and authentication
- access controls
- confidentiality
- integrity

User accounts, their rights and certificates are managed by Saint-Exupery/SystemX, allowing a restricted access (standalone mode of testbench), or an extended one (testbench inside the Confiance Environment).

REQ-00064: Cryptographic proof

To ensure the proper execution of it ML model, a cryptographic proof (e.g. : TLS certificates) should be integrated to resulted experiment performed.

■

REQ-00065: Cryptographic checksum

A cryptographic checksum shall be computed on the data and results used and produced by an experiment.

- Rationale: This elements performed on experiment packaging should be provided to ensure the integrity of environment built and the measures collected.

■

REQ-00066: Persistency of user data

No persistency of user's model and dataset should remain between experiments, except if the scheduling policy allow a user to keep the control on a target.

■

REQ-00067: Sharing of experiment definitions and results

The system shall provide the user with the capability to share <experiment definitions> and <experiment results> with other users in a discretionary manner.

■

B.3.3 Interoperability requirements

REQ-00068: Interoperability

The system shall be interoperable with the rest of the Confiance.AI environment.

■

Titre : ML Benchmarking Environment Specification

Mots clefs : Machine Learning, évaluation de performance

Ce document identifie les besoins en matière de moyens de *benchmarking* et d'optimisation du déploiement d'algorithmes de type *Machine Learning.*
Il définit la liste des exigences que le *banc d'évaluation d'algorithme ML* devra satisfaire.

Title : ML Benchmarking Environment Specification

Keywords : Machine learning, benchmarking

This document identifies the needs for a benchmarking and optimisation environment.
It defines the list of requirements with which this environment shall comply.

Our partners