Ref. Ares(2023)4409384 - 26/06/2023

Stairway to AI: Ease the Engagement of Low-Tech users to the AI-on-Demand platform through AI, H2020

Benchmarking results-2nd

Deliverable information	
Deliverable number	D6.4
WP number and title	WP6 - Vertical Matchmaking and hardware marketplace
Lead beneficiary	BCA
Dissemination level	Public
Due date	30 June 2023
Actual date of delivery	26 June 2023
Author(s)	Miguel de Prado (BCA), Andrea Borghesi (Unibo), Carmine Di Santi (Unibo), Jacopo Carletti (INFN), Hamdi Bouchech (HUA), Jagyan Prasad (HUA), Marco Rorro (EGI)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

Document Control Sheet

Version	Date	Summary of changes	Author(s)
0.1	11/01/2023	Initial draft of document	Miguel de Prado (BCA)
0.3	26/01/2023	First draft complete	Miguel de Prado (BCA), Andrea Borghesi (Unibo), Jacopo Carletti (INFN)
0.7	10/02/2023	Corrections after QA review	Miguel de Prado (BCA)
1.0	13/02/2023	Final document	Miguel de Prado (BCA)
1.3	25/05/2023	Additions to V1 version	Miguel de Prado (BCA), Andrea Borghesi (Unibo), Carmine Di Santi (Unibo), Jacopo Carletti (INFN), Hamdi Bouchech (HUA), Jagyan Prasad (HUA). Marco Rorro (EGI)
1.7	26/06/2023	Corrections after QA review	Miguel de Prado (BCA)
2.0	26/06/2023	Final document	Miguel de Prado (BCA)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

Table of contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

Acronyms

Acronym	Explanation
AI	Artificial Intelligence
API	Application Programming Interface
BMP	Bonseyes Marketplace
CPU	Central Processing Unit
DNN	Deep Neural Network
GPU	Graphical Processing Unit
HW	Hardware
LPDNN	Low-power Deep Neural Network framework
ML	Machine Learning
NPU	Neural Processing Unit
SW	Software
WP	Work Package

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

1. Executive Summary

The ***benchmark results*** **are the outcome of** ***benchmark software framework*** executed across multiple heterogenous platforms, which are then used **to train the vertical matchmaking engine** (Task 6.3).

In M25, the 1st version of the *Benchmark Results* was released within WP6 Task 6.2 of StairwAI project. The 1st version of *benchmark results* represented one of the main contributions from WP6 to train the 1st version of the vertical matchmaking engine (Task 6.3).

This document provides the 2nd version of the *Benchmark Results,* including the integration of Huawei's benchmarks that led the integration of Huawei's HW and SW into LPDNN (Benchmark Software Framework) within D6.2. Besides, this deliverable also provides a large step towards achieving MS10 for the integration of the Benchmark Results into the AI-on-demand platform by providing a database that can be accessed through the AI-on-demand API (Section 5).

This document is built incrementally on top of 1st version of the *Benchmark Results* deliverable, making the document self-contained. We add and highlight the new features and content of the *Benchmark Results 2nd* that have been added with respect to the 1st.

Section 2 introduces the deliverable, providing the context and the contributions.

Section 3 presents the results obtained from the *benchmark software framework* while Section 4 gives a summary of external benchmarks that are commonly used by researchers to push the limits of the fields.

In Section 5, we provide an analysis of these benchmarks and compare them against the ones collected within the project. We also propose a series of operational steps to integrate the benchmarking framework with StairwAI platform and with the AI-on-demand platform as well.

Lastly, Section 6 elaborates the conclusions.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

2. Introduction

As the computational capacity of embedded devices grows, they become more capable of performing sophisticated tasks, including those considered artificial intelligence (AI). Since the advent of AI for computer vision and speech recognition tasks, training of AI models has been performed using machine learning algorithms, with a large number of open-source datasets created by researchers. Such datasets provide an opportunity to develop, train and test AI algorithms on devices that range from the Cloud to the Edge.

A significant concern for AI engineers during the deployment of AI Algorithms is maximizing the system's performance in terms of overall latency, quality of solution, power, and space. The evaluation of AI Algorithms is often based on how well they perform on restricted public benchmark problems. These benchmarks give an informative indication of how well a given algorithm will perform on some given tested systems. However, AI Algorithms are often deployed on systems with rather different characteristics. As a result, the performance can vary widely according to the system on which they are deployed. Therefore, it is vital to have a large variety of benchmarks across multiple heterogeneous platforms to understand an AI Algorithm's behaviour comprehensively.

Taking these principles, the main contributions of this deliverable are the following:

- Presentation of benchmark results obtained by StairwAI's benchmark frameworks across heterogeneous platforms, including CPUs, GPUs, and NPUs.
- Survey of external benchmarks, presenting their scope, metrics, pros, and cons.
- Discussion about the results obtained towards creating a Vertical Matchmaking Engine and alignment with AI-on-demand platform.

The key additions with respect to the 1st version of the benchmarking results (D6.3) are the following:

- The addition of new datasets generated through the internal benchmarks by the project's partners.
- The identification of a common format to describe the output of the benchmarking results to make them usable by the vertical matchmaking engine.
- The creation of a data exchange service (deployed in the StairwAI platform) that acts as a connector between the benchmarking service and the vertical matchmaking service.
- The definition of a clear set of APIs for all services, to allow their integration into the AI-on-Demand.

2.1. Purpose and Scope of the document

WP6 has the main objective of building a Vertical Matchmaking engine that matches AI algorithms and HW resources to optimise the deployment of services and increase their efficiency. In the deliverable D6.2 (Benchmarking software-2nd version M28), the *benchmark software framework* was introduced, which had the main objectives of developing a software tool to benchmark machine learning models and hardware, including CPU (Intel x86, Arm Cortex-A5x, Risc-V), GPGPU (NVIDIA, etc), and NPU (HUA, etc) platforms.

Deliverable D6.4 is a Report, i.e., it introduces the 2nd version of the *Benchmark Results* collected by the *benchmark software framework*. Further, we offer the results of a survey we conducted to investigate existing benchmarking data, especially to understand whether it could be useful for the Vertical Matchmaking Engine (D6.5 in M30). Finally, we summarize the key insights gathered and provide some observations. We will explicitly focus the discussion on the aspects most relevant for the downstream task in WP6, namely the vertical matchmaking engine – which will exploit the benchmarking data to reach its goals. In addition, we also provide some key insights about the alignment with the AI-on-demand platform.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

3. Benchmark results (by StairwAI)

3.1. Benchmark Software Framework (LPDNN)

3.1.1. Scope

LPDNN is the deployment framework that was introduced in D6.1 and D6.2 as the *benchmark software framework* developed during T6.1. The LPDNN framework was initially developed during the H2020 Bonseyes project (Prado, Miguel De, et al) and has been largely extended in StairwAI to provide a structured benchmarking layer to analyze the execution of AI Applications on a variety of heterogeneous platforms. This layer, on top of LPDNN, adheres to the following design principles:

- ❑ Industry-driven Research:
  - Input requirements from SMEs wanting to use AI solutions.
  - Able to deploy AI solutions on edge devices (low-power, low-carbon footprint).
- ❑ Structured benchmarking workflow:
  - Availability of anchors within the deployment framework to evaluate the metrics truthfully.
  - Optimised deployment (value added to research and industry).
  - Extensive documentation & Support (user friendly for SMEs).
  - Defined interfaces (standardization).
  - Easy to replicate (reproducibility).
  - Create trust & community.

LPDNN provides the tools and capabilities to generate portable and efficient AI applications, which can be deployed and optimised across heterogeneous platforms, e.g., CPU, GPU, FPGA, NPU (ASIC). LPDNN features a full development flow for AI solutions on hardware devices by providing platform support, sample models, optimisation tools, integration of external libraries, and benchmarking.

LPDNN's full development flow makes the AI solution reliable and easy to replicate, increasing the **portability and fairness across the wide span of hardware platforms**. LPDNN's flexible architecture allows the main core to remain small and dependency-free. At the same time, additional 3rd party libraries or inference engines are only included when needed and for specific platforms, notably increasing the portability across systems. Thereby, different systems can be benchmarked with the same benchmarking code and fairly compared.

The collected dataset using LPDNN across heterogenous platforms has been open sourced at: https://gitlab.com/bonseyes/bonseyes-benchmarks/-/tree/master?ref_type=heads.

Over 3000 implementations have been benchmarked across 10 different HW platforms, allowing to train the Vertical Matchmaking engine (to be reported in D6.5 in M30).

For more details about LPDNN, please refer to D6.2.

3.1.2. Metrics

LPDNN provides a first analysis of static metrics during the offline compilation of the AI Applications. The offline metrics are the following:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

- **AI_TASK**: Task or class that the AI Model does, e.g., image classification.
- **MODEL_VERSION**: Specific version of the AI Model, e.g., different backbone.
- **INPUT_TILE**: Input size that the neural network takes.
- **PLATFORM**: HW platform where the AI Model is deployed.
- **ENGINE**: Inference engine that executes the AI Model.
- **PROCESSOR**: Processor where the AI Model is executed.
- **PRECISION**: Precision, e.g., floating point, of the data.
- **GFLOP**: Giga Floating Point OPerations that the AI Model's inference performs
- **#PARAMS**: Number of parameters (weights) that the AI Model contains.
- **STORAGE**: Size on disk of #PARAMS.

During the execution of the AI Applications, LPDNN measures the following metrics:

- **QUALITY_METRIC**: Quality metric, e.g., accuracy, mean square error.
- **QUALITY_VALUE**: Value of the quality metric.
- **PREPROCESSING_TIME**: Time spent on pre-processing the data.
- **INFERENCE_TIME**: Time spent on the forward pass of the neural network (NN).
- **POSTPROCESSING_TIME**: Time spent on post-processing the results of the NN.
- **LATENCY**: Addition of pre/post-processing+ inference times.
- **THROUGHPUT**: Number of executions per second. Inverse of latency.
- **DMIPS**: Dhrystone Million Instructions per Second.
- **CPU_MEM**: Average CPU memory allocated over the execution.
- **CPU_MEM_PEAK**: Peak CPU memory allocated over the execution.
- **GPU_MEM**: Average GPU memory allocated over the execution.
- **GPU_MEM_PEAK**: Peak GPU memory allocated over the execution.
- **MEMORY_BANDWIDTH**: Average Memory bandwidth taken.
- **MEMORY_BANDWIDTH**_PEAK: Peak Memory bandwidth taken.
- **CPU_LOAD**: Average load of the CPU over the execution.
- **GPU_LOAD**: Average load of the GPU over the execution.
- **NPU_LOAD**: Average load of the NPU over the execution.
- **CPU_TEMP**: Average temperature of the CPU over the execution.
- **GPU_TEMP**: Average temperature of the GPU over the execution.
- **POWER_CONSUMPTION**: Power consumption over the execution.
- **ENERGY_EFFICIENCY:** Energy efficiency over the execution.

For more detail about how these metrics are collected, please refer to D6.2.

Among these metrics, it is worth noting the different supported platforms, inference engines and AI models:

Platforms:

- **Raspberry Pi 3b+:** Quad-core ARM Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- **Raspberry Pi 4b:** Quad-core ARM Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- **Nvidia Jetson Nano**: Quad-core ARM Cortex -A57 @ 1.43 GHz & 128-core Nvidia Maxwell GPU
- **Nvidia Jetson Xavier:** Octa-core ARM v8.2 @ 2.03 GHz & 512-core Nvidia Volta GPU with Tensor Cores
- **Intel NUC:** Intel quad-core (TM) i5-7260U CPU @ 3.4 GHz

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

- **iMX8m Nano:** Quad-core ARM Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- **STM32 MP1:** Dual-core ARM Cortex-A7 cores up to @ 800 MHz
- **HPC**: General x86 Intel or AMD CPU cores and NVIDIAGPUs

As part of the 2nd release of the benchmarking results, Huawei's platforms have also been integrated and benchmarked:

- **Atlas 200 DK** (model: 3000): High-performance developer board that integrates the Ascend 310 AI processor. It has been widely used in scenarios such as developer solution verification, higher education, and scientific research.
- **Atlas 800 Training Server (Model: 9010)**: AI training server based on the Intel processors and Huawei Ascend 910 processors. It features ultra-high computing density and high network bandwidth. The server is widely used in deep learning model development and training scenarios and is an ideal option for computing-intensive industries, such as smart city, intelligent healthcare, astronomical exploration, and oil exploration.

Inference engines:

- **LNE**: LPDNN Native Engine (LNE) allows the execution of DNNs across arm-based and x86 CPUs as well as on NVIDIA-based GPUs.
- **NCNN**: NCNN ports the execution of DNNs on GPU through the Vulkan driver.
- **TensorRT**: TensorRT accelerates the DNN inference on NVIDIA-based GPUs and NPUs.
- **ONNXruntime**: ONNXruntime enables the direct execution of ONNX models on CPUs and GPUs.

As part of the 2nd release of the benchmarking results, Huawei's inference engine has also been integrated and benchmarked:

- **ACL**: AscendCL(ACL) enables the execution of DNNs on arm-based and x86 Huawei Ascend hardware.

AI Models:

- face-landmarks-detection-3ddfa
- body-pose-openpifpaf
- face-landmarks-detection-retinaface
- face-recognition-insightface-mobilefacenet
- incar-object-detection-nanodet
- age-gender-insightface
- emotion-classic-light-112x96
- eyegaze-rankgaze
- headpose-rankpose
- imagenet-alexnet
- imagenet-googlenet
- imagenet-efficientb0
- imagenet-squeezenet
- imagenet-mobilenet
- imagenet-mobilenet-v2
- imagenet-mobilenet-v3
- imagenet-resnet-resnet18

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

- imagenet-resnet-resnet32
- imagenet-resnet-resnet50
- coco-nanodet

3.1.3. Pros/Cons

PROS

- **Fair comparison across SW/HW vendors:**
  - Integration of SW inference engines from several vendors on the same codebase
  - Execution on heterogeneous systems (CPU, GPU, NPU)
- **Variety on Model deployment:**
  - Multiple flavours of models and backbones with several input sizes
  - Deployment with different inference engines and data types (fp32, fp16, int8)
- **Real Industrial benchmark**
  - Measures real latency of AI App with pre- & post-processing
  - Monitors system during benchmarking
  - Extensive number of metrics

CONS

- Mostly focused on deep learning model for computer vision
- Relatively small amount of data

3.2. Benchmark online optimization algorithms (Energy domain)

3.2.1. Scope

Online algorithms configuration for energy systems is a complex domain area characterized by uncertainty, e.g., renewable energy production, demand fluctuation, tight Hardware (HW) and real-time constraints. An online algorithm should be able to calculate the amount of energy that must be produced by the energy system to meet the required load, minimizing the total energy cost over the daily time horizon and by considering the uncertainty. This is a typical real-world problem with real time constraints.

Selecting the optimal HW configuration for a given set of tight constraints on solution time and quality over multiple and diverse data instances (e.g., the specific details of the energy system) for a given online algorithm is a complex problem. Moreover, each algorithm is characterized by a configurable parameter that further complicates this task.

The dataset has been collected using online algorithms and benchmarking software that complements the one presented in Sec 3.1 (LPDNN) for non-deep-learning algorithms. The dataset has been open sourced at https://zenodo.org/record/5838437.

For more details about the dataset, please refer to https://doi.org/10.1016/j.knosys.2022.109199

3.2.2. Metrics

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

The benchmark describes the behaviour of two different online algorithms with different parameter configurations and with different input instances (e.g., the specific details of the energy system and renewable energy production and demand fluctuation). In more detail, both algorithms are characterized by a single changeable parameter that impacts the measured metrics: algorithm runtime, memory consumption and solution quality.
For each record of the dataset, we collected the following information:

- **nParameter**: an integer that represents the number of scenario/traces used by the algorithm (the configurable parameter).
- **Load (kW)**: a vector of 96 values representing the load observations sampled in 96 stages (every 15 minutes over the course of a day).
- **RES (kW)**: a vector of 96 values representing the observations of available renewable production.

During the execution of the different algorithm configurations, we measure the following metrics:

- **solution quality (k\euro)**: a real number representing the value of the solution, in practice a measure of the daily total energy cost. It is obtained as the sum over the entire time horizon (over 96 stages) of all partial solutions.
- **runtime (sec)**: the time required for finding the solution with the algorithm. It is obtained as sum over the entire time horizon of all 96 partial running times.
- **memory consumption (MB)**: represents the RAM used by the algorithm on the machine where it was performed. It is an average of the memory used by the algorithm in 96 partial runs.

The dataset is generated by executing both the algorithms on different problem instances and with different parameter configurations.

Platform: All the runs for dataset creation have been performed on Intel Core i5 (3,1 GHz) machines with 16 GB of RAM.

3.2.3. Pros/Cons

CONS
The limitation of this dataset is mainly that we focused on a single HW architecture for analyzing the performance of both the algorithms with different parameter's configurations.

PROS
This dataset represents the first prototype to test the vertical Matchmaking (MM) engine:

- exhaustive grid-search exploration of the hyperparameter space
- exhaustive validation of the vertical MM engine on different configurations.

3.3. Benchmark complex, multi-parameter algorithms (Transprecision Computing Domain)

As part of the 2nd release of the benchmarking results, benchmarks on complex, multi-parameter algorithms (Transprecision Computing Domain) have also been performed.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

3.3.1. Scope

Transprecision computing[1] is a paradigm that allows users to trade the energy associated with computation in exchange for a reduction in the quality of the computation results. In this complex domain, a typical target is Floating-point (FP) operations: transprecision techniques allow to specify the number of bits used to represent FP variables, and using a smaller number of bits decreases the precision, thus saving energy. To analytically calculate the impact of varying the number of bits on the computation results for programs with more than a couple of instructions is a crucial point. However, this relationship can be learned from data.

To execute the transprecision benchmarks we used an open-source framework called *Flex-Float*[2] that allows for the emulation of algorithms' execution at different FP precisions and benchmarking software as explained in Sec 3.2. A comprehensive description of the framework is outside the scope of this document but can be found in the work of [Tagliavini et al., 2018][3].

We consider numerical benchmarks where multiple FP variables take part in the computation of the result for a given input set, which includes a structured set of FP values (typically a vector or a matrix); the precision assigned to the different FP variables can be changed by running the benchmark under different configurations. Assigning a precision means deciding the number of bits for the mantissa; the exponent dictates the extension of dynamic range and is set according to the actual types available on the target HW platform. We refer to [Borghesi et al., 2020][4] for additional details on the problem of running transprecision computing benchmarks while assigning different precisions to the FP variables. For our purposes, the important thing to understand are the differences between the transprecision domain and the energy domain described in Section 3.2:

- In the transprecision domain we consider a set of four different algorithms, namely a subset of the applications studied in the context of transprecision computing, chosen because they represent distinct problems and capture different patterns of computation – in the energy domain case we considered two completely different algorithms. In particular, we considered the following four algorithms:
    - *FWT*, Fast Walsh Transform for real vectors, from the domain of advanced linear algebra; the number of FP variables is 2.
    - *Saxpy*, a generalized vector addition (basic linear algebra), with a number of FP variables equal to 3.
    - *Convolution*, the convolution of a matrix with a 11x11 kernel; 4 FP variables are needed by this algorithm.
    - *Correlation, to* compute the correlation matrix given as input, with a number of FP variables equal to 7.

[1] Malossi, A. Cristiano I., et al. "The transprecision computing paradigm: Concept, design, and applications." *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018.
[2] https://github.com/oprecomp/flexfloat
[3] Tagliavini, Giuseppe, Andrea Marongiu, and Luca Benini. "Flexfloat: A software library for transprecision computing." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39.1 (2018): 145-156.
Tagliavini, Giuseppe, et al. "A transprecision floating-point platform for ultra-low power computing." *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018.
[4] Borghesi, Andrea, et al. "Combining learning and optimization for transprecision computing." *Proceedings of the 17th ACM International Conference on Computing Frontiers*. 2020.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

- While in the energy domain the algorithms' behaviour was governed by a single hyperparameter (a different parameter for each of the two algorithms), in the transprecision case the hyperparameters involved correspond to the number of FP variables in the selected algorithms.
    - Different runs of the algorithms were done with different configurations of hyperparameters, that is the algorithms were run with different values of bits assigned to the FP variables (ranging from 4 to 53, bounds identified on the basis of domain knowledge).
    - The hyperparameter space was explored through Latin Hypercube Sampling.
    - For each hyperparameter configuration, 30 different runs were performed, each characterized by a different input instance (some of metrics describing the behaviour of the algorithm are impacted by the input – e.g., the initial vector or matrix – fed to the algorithms).
- The benchmarks were run on three different hardware platforms, whereas in the energy domain case a single HW platform was used.
    - All hyperparameter configurations were run on the three HW devices.
    - The different HW platforms are the following:
        - A consumer-grade laptop
        - A virtual Machine (with dedicated usage)
        - A High-Performance Computing node

3.3.2. Metrics

During the execution of the different algorithm configurations, we measure the following metrics:

- **computation error (absolute number)**: a real number representing the "quality" of the solution, in practice the ration of the execution of the transprecision computing algorithms with the FP variables at reduced number of bits and the execution of the same algorithms with all the variables at maximum number of bits.
- **runtime (sec)**: the time required for finding the solution with the algorithm.
- **memory consumption (MB)**: represents the RAM used by the algorithm on the machine where it was performed.

3.3.3. Pros/Cons

CONS

The key disadvantage of the transprecision benchmark lies in the complexity of the domain: it is notoriously difficult to learn any function trying to characterize the behavior of transprecision algorithms[5], even when keeping the HW resources fixed. This problem is clearly only compounded when considering multiple HW platforms.

PROS

The main advantage of this suite of benchmarks (at least compared to the case of online algorithms for the energy domain) is that different HW platforms have actually been used, hence it is possible to directly observe the behavioral changes due to the impact of the hardware.

[5] Borghesi, Andrea, et al. "Combining learning and optimization for transprecision computing." *Proceedings of the 17th ACM International Conference on Computing Frontiers*. 2020.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

4. Survey of external benchmarks

In addition to the benchmarks performed internally under StairwAI, we also offer the results of a survey we conducted to investigate existing external benchmarking data, especially to understand whether it could be useful for the Vertical Matchmaking Engine (D6.5 in M30). We analyse and gather multiple datasets, which we present below:

4.1. AI Benchmark

4.1.1. Scope

The AI Benchmark is an Android application designed to check the performance and the memory limitations associated with running AI and deep learning algorithms on mobile platforms. It consists of several computer vision tasks performed by neural networks that are running directly on Android devices. The considered networks represent the most popular and commonly used architectures that can be currently deployed on smartphones.

Besides the Android version, a separate open-source AI Benchmark build for desktops was released in June 2019. It is targeted at evaluating AI performance of the common hardware platforms, including CPUs, GPUs and TPUs, and measures the inference and training speed for several key deep learning models. The benchmark is relying on the TensorFlow machine learning library and is distributed as a Python pip package that can be installed on any system running Windows, Linux or macOS.

Reference Paper: https://arxiv.org/pdf/1810.01109.pdf, https://arxiv.org/pdf/1910.06663.pdf

Website: https://ai-benchmark.com/index.html

4.1.2. Metrics

Supported Mobile Architectures:

- Qualcomm: Snapdragon 845 (Hex. 685 + Adreno 630); Snapdragon 710 (Hexagon 685); Snapdragon 670 (Hexagon 685); Snapdragon 855+ (Hex. 690 + Adreno 640); Snapdragon 855 (Hex. 690 + Adreno 640); Snapdragon 730 (Hex. 688 + Adreno 618); Snapdragon 675 (Hex. 685 + Adreno 612); Snapdragon 665 (Hex. 686 + Adreno 610).
- HiSilicon: Kirin 970 (NPU, Cambricon); Kirin 980 (NPU×2, Cambricon).
- Samsung: Exynos 9810 (Mali-G72 MP18); Exynos 9610 (Mali-G72 MP3); Exynos 9609 (Mali-G72 MP3); Exynos 9825 (NPU + Mali-G76 MP12); Exynos 9820 (NPU + Mali-G76 MP12).
- MediaTek: Helio P70 (APU 1.0 + Mali-G72 MP3); Helio P60 (APU 1.0 + Mali-G72 MP3); Helio P65 (Mali-G52 MP2; Helio P90 (APU 2.0); Helio G90 (APU 1.0 + Mali-G76 MP4).

Frameworks:

- TensorFlow Mobile
- TensorFlow Lite
- Caffe2

Deep Learning Tests:

- **Test Section 1: Image Classification**, Model: MobileNet-V2, Inference modes: CPU (FP16/32) and NNAPI (INT8 + FP16), Image resolution: 224×224 px, Test time limit: 20 seconds.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

- **Test Section 2: Image Classification**, Model: Inception-V3, Inference modes: CPU (FP16/32) and NNAPI (INT8 + FP16), Image resolution: 346X346 px, Test time limit: 30 seconds.
- **Test Section 3: Face Recognition**, Model: Inception-ResNet-V1, Inference modes: CPU (INT8) and NNAPI (INT8 + FP16), Image resolution: 512X512 px, Test time limit: 30 seconds.
- **Test Section 4: Playing Atari**, Model: LSTM, Inference modes: CPU (FP16/32), Image resolution: 84X84 px, Test time limit: 20 seconds.
- **Test Section 5: Image Deblurring**, Model: SRCNN 9-5-5, Inference modes: NNAPI (INT8 + FP16), Image resolution: 384X384 px, Test time limit: 30 seconds.
- **Test Section 6: Image Super-Resolution**, Model: VGG-19 (VDSR), Inference modes: NNAPI (INT8 + FP16), Image resolution: 256X256 px, Test time limit: 30 seconds.
- **Test Section 7: Image Super-Resolution**, Model: SRGAN, Inference modes: CPU (INT8 + FP16/32), Image resolution: 512X512 px, Test time limit: 40 seconds.
- **Test Section 8: Bokeh Simulation**, Model: U-Net, Inference modes: CPU (FP16/32), Image resolution: 128X128 px, Test time limit: 20 seconds.
- **Test Section 9: Image Segmentation**, Model: ICNet, Inference modes: NNAPI (2 X FP32 models in parallel), Image resolution: 768X1152 px, Test time limit: 20 seconds.
- **Test Section 10: Image Enhancement**, Model: DPED-ResNet, Inference modes: NNAPI (FP16 + FP32), Image resolution: 128X192 px, Test time limit: 20 seconds.
- **Test Section 11: Memory Test**, Model: SRCNN 9-5-5, Inference modes: NNAPI (FP16), Image resolution: from 200X200 px to 2000X2000 px.

Desktop GPU and CPUs architectures:

- Tesla V100 SXM2 32Gb
- Tesla V100 PCIE 32Gb
- NVIDIA Quadro GV100
- NVIDIA Quadro RTX 8000
- GeForce RTX 2070 SUPER
- NVIDIA TITAN Xp CE
- AMD Radeon VII
- GeForce RTX 2080 Max-Q
- GeForce RTX 2060 Laptop
- NVIDIA Tesla T4
- Intel Xeon Gold 6148
- Intel Xeon Gold 6248
- AMD EPYC 7451

Desktop GPU and CPUs datasets and tasks

- MobileNet-V2 [classification]
- Inception-V3 [classification]
- Inception-V4 [classification]
- Inception-ResNet-V2 [classification]
- ResNet-V2-50 [classification]
- ResNet-V2-152 [classification]
- VGG-16 [classification]

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

- SRCNN9-5-5 [image-to-image mapping]
- VGG-19 [image-to-image mapping]
- ResNet-SRGAN [image-to-image mapping]
- ResNet-DPED [image-to-image mapping]
- U-Net [image-to-image mapping]
- Nvidia-SPADE [image-to-image mapping]
- ICNet [image segmentation]

Scoring System

AI Benchmark is measuring the performance of several test categories, including int-8, float-16, float-32, parallel, CPU (int-8 and float-16/32), memory tests, and tests measuring model initialization time.

The contribution of the test categories is as follows:

- 48%-float-16 tests;
- 24%-int-8 tests;
- 12%-CPU,float-16/32 tests;
- 6%-CPU,int-8 tests;
- 4%-float-32 tests;
- 3%-parallel execution of the models;
- 2%-initialization time, float models;
- 1%-initialization time, quantized models;

The scores of each category are computed as a geometric mean of the test results belonging to this category. The computed L1 error is used to penalize the runtime of the corresponding networks running with NNAPI (an exponential penalty with exponent 1.5 is applied).

4.1.3. Pros/Cons

PROS

- **Heavy focus on computer vision**
- **Multiple ML architecture**:
  - MobileNet, Inception
- **Multiple precision for the same ML architecture**:
  - FP16, INT8
- **State of Art Platforms/hardware**:
  - AMD processors, NVIDIA accelerators, etc
- **Industry standard and unbiased evaluations**:
  - Recognised by major companies and universities.
- **Regularly updated with new platforms**

CONS

- **Mobile Oriented:** CPU and GPU results are not easy to compare with Mobile Inference. Still under development

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

- **Relies on TensorFlow Lite:** the number of critical bugs and issues introduced in its new versions prevents from recommending it for any commercial projects or projects dealing with non-standard AI models

4.2. MLPerf

4.2.1. Scope

MLPerf aims to create a representative benchmark suite for ML that evaluates system performance to meet five high-level goals:

1. Enable fair comparison of competing systems while still encouraging ML innovation.
2. Accelerate ML progress through fair and useful measurement.
3. Enforce reproducibility to ensure reliable results.
4. Serve both the commercial and research communities.
5. Keep benchmarking effort affordable so all can participate.

Reference Paper: https://arxiv.org/abs/1911.02549

Website: https://mlcommons.org/en/, section “Benchmarks”, subsections “Training” and “Inference”

Training

MLPerf Training does the following:

1. Establish a comprehensive benchmark suite that covers diverse applications, DNN models, and optimizers.
2. Create reference implementations of each benchmark to precisely define models and training procedures.
3. Establish rules that ensure submissions are equivalent to these reference implementations and use equivalent hyperparameters.
4. Establish timing rules to minimize the effects of stochasticity when comparing results.
5. Make submission code open source so that the ML and systems communities can study and replicate the results.
6. Form working groups to keep the benchmark suite up to date.

ML areas, including vision, language, recommendation, and reinforcement learning set of seven benchmarks.

Inference

ML inference systems range from deeply embedded devices to smartphones to data centres. They have a variety of real-world applications and many figures of merit, each requiring multiple performance metrics. The right metrics, reflecting production use cases, allow not just MLPerf but also publications to show how a practical ML system would perform. MLPerf Inference consists of four evaluation scenarios: single-stream, multistream, server, and offline.

These scenarios represent many critical inference applications. MLPerf Inference provides a way to simulate the realistic behaviour of the inference system under test.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 101017142

4.2.2. Metrics

Training

Area	Benchmark	Dataset	Quality Threshold	Model
Vision	Image classification	ImageNet	75.90% classification	ResNet-50 v1.5
Vision	Image segmentation (medical)	KiTS19	0.908 Mean DICE score	3D U-Net
Vision	Object detection (light weight)	Open Images	34.0% mAP	RetinaNet
Vision	Object detection (heavy weight)	COCO	0.377 Box min AP and 0.339 Mask min AP	Mask R-CNN
Language	Speech recognition	LibriSpeech	0.058 Word Error Rate	RNN-T
Language	NLP	Wikipedia 2020/01/01	0.72 Mask-LM accuracy	BERT-large
Commerce	Recommendation	1TB Click Logs	0.8025 AUC	DLRM
Research	Reinforcement learning	Go	50% win rate vs. checkpoint	Mini Go (based on Alpha Go paper)

Training Supported Hardware

- NC96ads_A100_v4
- ND96amsr_A100_v4_n16
- ND96amsr_A100_v4_n8
- ESCN4A-E11
- ESC8000A-E11-8xA100-PCIE-80GB-NVBridge
- 8_node_64_A100_PaddlePaddle
- R750xax4A100-PCIE-80GB
- PRIMERGY-RX2540M6-mxnet
- 8xR750xax4A100-PCIE-80GB
- G492-ZD2
- HPE-ProLiant-XL675d-Gen10-Plus_A100-SXM-80GB_hugectr
- HLS-Gaudi2-PT
- Dell Precision 7920 Tower with 2x A5000 using MxNet 22.04
- Lenovo ThinkSystem SR670 V2 Server with 4x 40GB SXM4 A100
- Lenovo ThinkSystem SR670 V2 Server with 8x 80GB PCIe A100
- dgxa100_ngc22.04_merlin_hugectr
- AS-4124GS-TNR

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

- G5500V6x8xA30
- 1-node-SPR-pytorch
- 16-nodes-SPR-pytorch
- dgxh100_n4_preview
- NF5468M6J

Training HPC

Area	Benchmark	Dataset	Quality Threshold	Model
Scientific	Climate segmentation	CAM5+TECA simulation	IOU 0.82	DeepCAM
Scientific	Cosmology parameter prediction	CosmoFlow N-body simulation	Mean average error 0.124	CosmoFlow
Scientific	Quantum molecular modeling	Open Catalyst 2020 (OC20)	Forces mean absolute error 0.036	DimeNet++

Time-to-Train Performance Metric

To address the ML-benchmarking challenges of system optimization and scale, MLPerf performance metric is the time to train to a defined quality target.

It incorporates both system speed and accuracy and is most relevant to ML practitioners. As an end-to end metric, it also captures the auxiliary operations necessary for training such models, including data-pipeline and accuracy calculations. The metric's generality enables application to reinforcement learning, unsupervised learning, generative adversarial networks, and other training schemes.

Each benchmark measures the wall clock time required to train a model on the specified dataset to achieve the specified quality target. To account for the substantial variance in ML training times, final results are obtained by measuring the benchmark a benchmark-specific number of times, discarding the lowest and highest results, and averaging the remaining results. Even the multiple result average is not sufficient to eliminate all variance. Imaging benchmark results are very roughly +/- 2.5% and other benchmarks are very roughly +/- 5%.

For non-HPC training, results that converged in fewer epochs than the reference implementation run with the same hyperparameters were normalized to the expected number of epochs.

Inference

MLPerf defines model-quality targets. We established per-model and scenario targets for inference latency and model quality. The latency bounds and target qualities are based on input gathered from ML-system end users and ML practitioners. As MLPerf improves these parameters in accordance with industry needs, the broader research community can track them to stay relevant.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

Inference Datacenter

Area	Benchmark	Dataset	Quality Threshold	Model
Vision	Image classification	ImageNet (224x224)	99% of FP32 (76.46%)	Resnet50-v1.5
Vision	Object detection	OpenImages (800x800)	99% of FP32 (0.20 mAP)	Retinanet
Vision	Medical image segmentation	KITS 2019 (602x512x512)	99% of FP32 and 99.9% of FP32 (0.86330 mean DICE score)	3D UNET
Speech	Speech-to-text	Librispeech dev-clean (samples < 15 seconds)	99% of FP32 (1 - WER, where WER=7.452253714852645%)	RNNT
Language	Language processing	SQuAD v1.1 (max_seq_len=384)	99% of FP32 and 99.9% of FP32 (f1_score=90.874%)	BERT-large
Commerce	Recommendation	1TB Click Logs	99% of FP32 and 99.9% of FP32 (AUC=80.25%)	DLRM

Metrics:

A. queries/s
B. samples/s
C. Accuracy
D. System Power (W)

Inference Mobile

Area	Benchmark	Dataset	Quality Threshold	Model
Vision	Image classification	ImageNet	98% of FP32 (Top1: 76.19%)	MobileNetEdgeTPU
Vision	Object detection	MS-COCO 2017	95% of FP32 (mAp: 0.285)	MobileDETs
Vision	Segmentation	ADE20K (32 classes, 512x512)	97% of FP32 (32-class mIOU: 54.8)	DeepLabV3+ (MobileNetV2)
Vision	Segmentation, MOSAIC	ADE20K (32 classes, 512x512)	96% of FP32 (32-class mIOU: 59.8)	MOSAIC
Language	Language processing	SQUAD 1.1	93% of FP32 (F1 score: 90.5)	Mobile-BERT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

Metrics:

A. frames/s
B. latency in ms

4.2.3. Pros/Cons

PROS

The framework offers different sorts of benchmarks for comparison/innovation:

- **Close:** Same original model on HW or SW framework
- **Open:** - Allow to innovate with networks and frameworks
- **Power:** - Allows to obtain system power measurements
- **Scenarios:** - Different streams: single, multi, server, offline
- **Platforms:** - Many HW platforms and SW frameworks

CONS

- Only few tasks
- Only few models
- No memory usage
- No system usage
- No pre- or post-processing

4.3. AIBench

4.3.1. Scope

AIBench provides a scalable and comprehensive data-center AI benchmark suite. In total, it includes 12 micro benchmarks, 16 component benchmarks, covering 16 AI problem domains: image classification, image generation, text-to-text translation, image-to-text, image-to-image, speech-to-text, face embedding, 3D face recognition, object detection, video prediction, image compression, recommendation, 3D object reconstruction, text summarization, spatial transformer, learning to rank, and two end-to-end application. AI benchmarks:

- DCMix —a datacenter AI application combination mixed with AI workloads
- E-commerce AI—an end-to-end business AI benchmark.

Reference Paper: https://arxiv.org/pdf/2004.14690, http://www.benchcouncil.org/aibench/file/AIBench-Bench18.pdf, https://www.benchcouncil.org/file/Cluster_2021_hpcai_camera.pdf

Website: https://www.benchcouncil.org/aibench/index.html, section “AIBench Training” and “AIBench Inference”

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 101017142

4.3.2. Metrics

AI Tasks

- **Image generation** uses WGAN algorithms and uses LSUN dataset as data input to generate image data.
- **Text-to-Text Translation** uses recurrent neural networks and takes WMTEnglish-German as data input to translate text data.
- **Image-to-Text** uses Neural Image Caption model and takes Microsoft COCO dataset as input to describe image using text.
- **Image-to-Image** uses the cycleGAN algorithm and takes Cityscapes dataset as input to transform the image to another image.
- **Speech-to-Text** uses the DeepSpeech2 algorithm and takes Librispeech dataset as input to recognize the speech data.
- **Face embedding** uses the FaceNet algorithm and takes the LFW (Labeled Faces in the Wild) dataset or VGGFace2 as input to convert image to an embedding vector.
- **3Dface recognition** uses 3D face modes to recognize 3D information within images. The input data includes 77,715 samples from 253 face IDs, which is published on the BenchCouncil web site.
- **Object detection** uses the Faster R-CNN algorithm and takes Microsoft COCO dataset as input to detect objects in images.
- **Recommendation** uses collaborative filtering algorithm and takes MovieLens dataset as input to provide recommendations.
- **Video prediction** uses motion-focused predictive models and takes Robot pushing dataset as input to predict video frames.
- **Image compression** uses recurrent neural networks and takes ImageNet dataset as input to compression images.
- **3D object reconstruction** uses a convolutional encoder-decoder network and takes ShapeNet Dataset as input to reconstruct 3D object.
- **Text summarization** uses sequence-to-sequence model and takes Gigaword dataset as input to generate summary description for text.
- **Spatial transformer** uses spatial transformer networks and takes MNIST dataset as input to make spatial transformations.
- **Learning to Rank** uses ranking distillation algorithm and uses Gowalla dataset to generate ranking scores.

Metrics

- Wall Clock Time
- Energy consumption of running a benchmark
- Accuracy

Provides both training and inference benchmarks.

The training metrics are the wall clock time to train the specific epochs, the wall clock time to train a model achieving a target accuracy, and the energy consumption to train a model achieving a target accuracy.

The inference metrics are the wall clock time, accuracy, and energy consumption.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

Additionally, the performance numbers are reported on the BenchCouncil web site (http://www.benchcouncil.org/numbers.html), to measure the training and inference speeds of different hardware platforms, including multiple types of NVDIA GPUs, Intel CPUs, AI accelerator chips, and to measure the performance of different software stacks, including TensorFlow, PyTorch, etc.

4.3.3. Pros/Cons

PROS

- **Wide variety of data types** and data sources are covered, including text, images, street scenes, audios, videos, etc.
- **Not only based on mainstream deep learning frameworks** like TensorFlow and PyTorch, but also based on traditional programming model like Pthreads.
- Provides Training, Inference, Micro and Synthetics Benchmarks across Datacenter, HPC, IoT, and Edge.

CONS

- Last update in 2021
- **Lacking benchmark results** on latest Mobile and CPU/GPUs architectures

4.4. OpenML CC-18

4.4.1. Scope

Seamlessly integrated into the OpenML platform, this benchmark suites standardize the setup, execution, analysis, and reporting of benchmarks:

- All datasets are uniformly formatted in standardized data formats.
- They can be easily downloaded programmatically through APIs and client libraries.
- They come with machine-readable meta-information, such as the occurrence of missing values, to train algorithms correctly.
- Standardized train-test splits are provided to ensure that results can be objectively compared.
- Results can be shared in a reproducible way through the APIs.
- Results from other users can be easily downloaded and reused.

Reference Paper: https://arxiv.org/pdf/1708.03731.pdf

Website: https://www.openml.org/search?type=benchmark&sort=tasks_included&study_type=task, Benchmarking Doc: https://docs.openml.org/benchmark/

4.4.2. Metrics

The OpenML-CC18 contains all verified and publicly licenced OpenML datasets until mid-2018 that satisfy a large set of clear requirements for thorough yet practical benchmarking:

- The number of observations is between 500 and 100000 to focus on medium-sized datasets that can be used to train models on almost any computing hardware.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

- The dataset has less than 5000 features, counted after one-hot-encoding categorical features (which is the most frequent way to deal with categorical variables), to avoid most memory issues.
- The target attribute has at least two classes, with no class of less than 20 observations. This ensures sufficient samples per class per fold when running 10-fold cross-validation experiments.
- The ratio of the minority and majority class is above 0.05 (to eliminate highly imbalanced datasets which require special treatment for both algorithms and evaluation measures).
- The dataset is not sparse because not all machine learning models can handle them gracefully, this constraint facilitates our goal of wide applicability.
- The dataset does not require taking time dependency between samples into account, e.g., time series or data streams, as this is often not implemented in standard machine learning libraries. Removed datasets where each sample constitutes a single data stream.
- The dataset does not require grouped sampling. Such datasets would contain multiple data points for one subject and require that all data points for a subject are put into the same data split for evaluation.

4.4.3. Pros/Cons

PROS

- **Easy creation of benchmarks**
- **Permanence and provenance**: Because benchmarking suites are its own entity on OpenML, it is clear who created them (provenance).
- **Community of practice**: Curated benchmark suites allow scientists to thoroughly benchmark their machine learning methods without having to worry about finding and selecting datasets for their benchmarks.
- **Building on existing suites**: Scientists can extend, subset, or adapt existing benchmarking suites to correct issues, raise the bar, or run personalized benchmarks.
- **Reproducibility of benchmarks**: Based on machine-readable OpenML tasks, with detailed instructions for evaluation procedures and train-test splits, shared results are comparable and reproducible.

CONS

- **Overfitting**: overfitting on fixed suites is increasingly likely.
- **Computational issues**: focused on mid-size datasets, some larger ones still incurred too high computational load, so some researchers have used subsets of the OpenML-CC18 in their work.
- **Breadth of current benchmarking suites**: researchers are interested in benchmarking larger (deep learning) models on larger datasets from many domains (including language and vision).
- **Specification of resource constraints**: the task and suite specifications do not yet allow for constraints on resources, e.g., memory or time limits.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

4.5. DataPerf

4.5.1. Scope

DataPerf focuses on creating better datasets than creating better models regarding the breadth, difficulty, and faithfulness of datasets employed in ML tasks. It is designed to improve the training and test data for model benchmarking and, consequently, ML models. The DataPerf benchmark suite is a collection of tasks, metrics and rules to understand the scope, quality and limitations of datasets. It measures the quality of training and test datasets and the quality of algorithms for constructing such datasets. Benchmark examples include data debugging, data valuation, training- and test-set creation, and selection algorithms. Currently, speech and vision benchmarks are designed to evaluate dataset creation and selection.

DataPerf has the following goals:

- Focus research and development on improving ML dataset quality.
- Improve ML training datasets to increase accuracy and/or reduce data required to train.
- Improve ML test datasets to drive ML solution fidelity and reliability.
- Motivate datasets that increase representation and decrease bias.
- Drive development of better techniques and tools for creating and optimizing datasets.
- Provide consistent metrics for researchers and commercial developers.
- Enforce replicability to ensure reliable results.
- Keep benchmarking effort affordable so all can participate.

Reference Paper: https://arxiv.org/pdf/2207.10062

Website: https://dataperf.org/

4.5.2. Metrics

The DataPerf suite includes the benchmark types listed below. Each benchmark type uses a different metric, though all in principle either maximize the efficacy of a training set or the breadth/difficulty of a test set.

DataPerf Benchmark type

Benchmark Type	Benchmark Method	Benchmark Metric
Training Set Creation	Replace given training set with novel training set	Accuracy of models trained on novel training set
Test Set Creation	Select a fixed number of additional test-data items from the supplemental set	OpenI Number of submitted test data items incorrectly labeled by models and correctly labeled by humans, where credit for each item is divided by number of

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

		submissions containing that item
Selection Algorithm	Select a fixed number of additional test-data items from the supplemental set	Number of submitted test data items incorrectly labeled by models and correctly labeled by humans, where credit for each item is divided by number of submissions containing that item
Debugging Algorithm	Identify labeling errors in version of training set that contains some corrupted labels	Accuracy of trained models after identified labels are corrected
Slicing Algorithm	Divide training set into semantically coherent slices	Fraction of data assigned to the correct slice
Valuation Algorithm	Estimate accuracy improvement from training on set A to training on set A + set B (where B lacks labels at time of estimate)	Absolute difference between predicted accuracy and actual accuracy

4.5.3. Pros/Cons

PROS

- Focused on Data
- Improved Data Quality

CONS

- Under Development, few results available
- Mainly focused on accuracy, independently of the hardware
- No HW metrics available

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

5. Discussion and integration

5.1. Benchmarks discussion

Multiple insights could be gained from the survey of available datasets and benchmarks presented in the previous sections. In particular, we will consider the relation of the benchmarks with the final **goal of the WP6**, which is to **build a vertical matchmaking component** capable of matching available hardware resources and AI algorithms while respecting user-specified constraints. One of the critical components of the vertical engine is a Machine Learning model, which learns the relation between the behaviour of the algorithm (for instance, represented as time-to-solution, runtime, memory consumption, power consumption, and other similar metrics) and the HW platform used to run the algorithm.

From the point of view of the vertical matchmaking engine, **most of the value** of the benchmarking data **lies in** the possibility of **characterizing the behaviour of AI algorithms** running across different hardware platforms and under different configurations. The main key to be drawn is to understand whether the deployment of the algorithm can be maximized by being deployed on a specific platform or configuration. The vertical matchmaker will then use this information to support the user in selecting the optimal hardware resources for a given task, i.e., a specific algorithm, and possibly to find the optimal hyperparameters' configuration.

The **benchmarks provided by StairwAI**, e.g., LPDNN's dataset, have into consideration the support of multiple platforms and configurations, e.g., inference engine, model version, model size, data type, etc., **addressing those vertical matchmaking engine's requirements** explained above. Thus, the **first version of vertical matchmaking has already been trained** on these datasets, which will be further described in D6.5.

However, this information is not always explicitly reflected in **the external benchmarking datasets**, which are **not always created to characterize the algorithms' behaviour**, i.e., they tend to either collect information about a few algorithms executed on multiple platforms or a modest set of algorithm's configurations on a single device. This does not necessarily make these data worthless, as helpful information can be extracted, especially for transfer learning purposes.

It can also be noticed that there is **no general unified format employed** to collect data and perform the benchmarks. This lack of a standard **hinders the development of a vertical matchmaking** engine over different domains, as an ad-hoc technique to process the benchmarking data would need to be implemented for each case. A standard and unified format could be highly beneficial in this regard – it could be as simple as deciding a common format by which the collected data should be organized. For instance, the first step could consist in creating a companion meta-data descriptor containing the information about the benchmarking data and easily readable by a machine for automated processing. We further develop this concept in Section 5.2.2.

Another observation that can be made is the fact that existing benchmarking datasets cover a wide range of different target metrics, that is, the metrics used to characterize the algorithms' behaviour and measured during the execution on a specific HW platform. On the one hand, the **variety of target metrics** is a boon as it allows for studying an algorithm's behaviour under different aspects. On the other hand, this might **limit the possibility of comparing** different algorithms and/or different HW devices, as the comparison is difficult when the benchmarks measure different things. Further, fairness can be at stake if the benchmarking process across the various frameworks is not performed homogeneously.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

Overall, and after careful consideration, we have realized that it is extremely difficult to use external benchmarks in their current form to train the vertical matchmaking engine since these benchmarks, while very informative, were not explicitly devised to support the downstream task that we are considering in this work package. Most of the benchmarks only contain information about the runtime (and not always). Thus, it is challenging to properly characterize the algorithms' behaviour in other aspects, strongly limiting the benefits of using the vertical matchmaking engine. Moreover, although data from several platforms and algorithms are provided, the benchmarks do not explore the configuration space in an exhaustive way, which is an aspect essential to building data-driven models to fit the algorithms' behaviour. Finally, since no general unified format is employed, an ad-hoc technique to process the benchmarking data would need to be implemented for each case, which would require an unfeasible amount of work for this project.

For these reasons, we have decided to **rely** entirely **on internally generated benchmarks** - which provide a **richer configuration space** and address the **heterogeneity** of HW devices and configurations - as inputs **for the vertical matchmaking engine** as will be extensively presented in D6.5.

5.2. Benchmark Integration

As part of the 2nd release of the benchmarking results, the benchmark integration has been widely addressed. In this Section, we introduce the StairwAI's benchmark services and how they align with the AI-on-demand. Figure 1 illustrates, from a user's perspective, the benchmarking workflow:

A. The user gets an algorithm that is fetched from the AI-on-demand.
B. The fetched algorithm is benchmarked with the Benchmark as a Service (aaS).
C. The benchmark is stored in the Data exchange Service.
D. The Vertical Matchmaking Engine uses the Data Exchange Service to train the engine on the available benchmark data.
E. The user may specify his/her constrains to the Vertical Matchmaking Engine and gets a supportive recommendation (prediction) for a given unbenchmarked algorithm or platform.
F. The user push benchmark results to the AI-on-demand.

In the following sub-sections, we ellaborate on each of the presented components.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

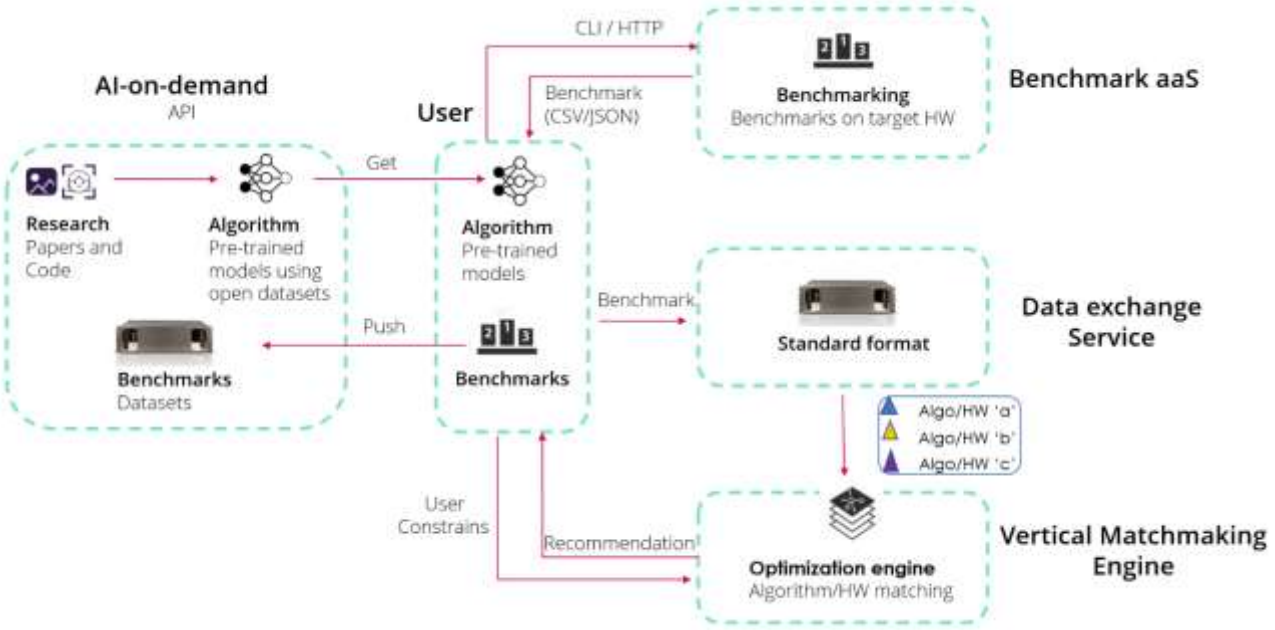

Figure 1 StairwAI's Benchmarking

5.2.1. Benchmark as a Service

In D6.2, the Benchmark as-a-Service's (BaaS) design was introduced in collaboration with WP2 for the broader interoperability with the AI-on-demand platform as shown in Figure 1. StairwAI aims to enhance the AI-on-demand by adding a service layer that provides, among others, a Benchmark aaS. The Benchmark aaS provides a service layer wrapping and also extending the *Benchmarking Software framework* (D6.2), alleviating some of the *Benchmarking software framework* dependencies and be able to benchmark any AI model. Figure 2 shows how a user can interact with the Benchmark aaS to benchmark an AI model on a variety of platforms, using different configurations.

For more details about the benchmark aaS, please refer to D6.2

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

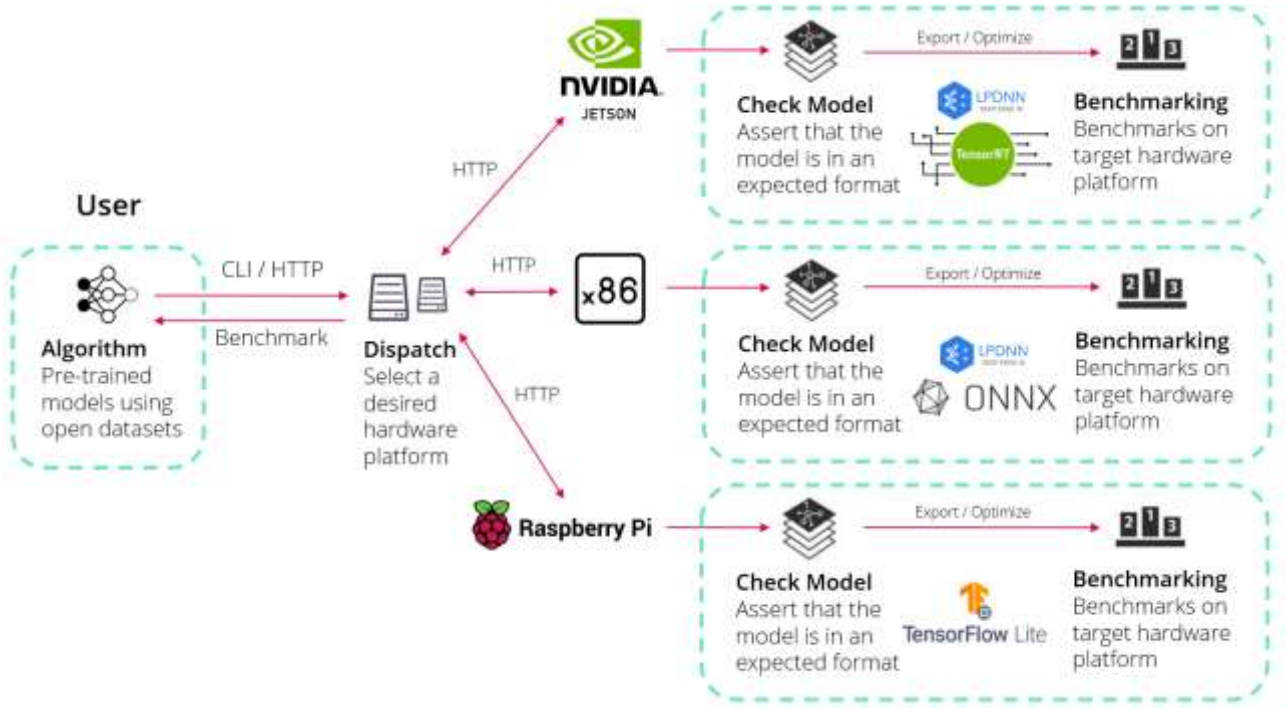

Figure 2 Benchmark aaS

5.2.2. Metadata describing the benchmarking output

A fundamental step to obtain reusable and understandable benchmarks, as discussed in Sec 5.1, is to endow them with all the necessary meta-data, such as the specifics of the executed AI asset, the HW device where it was deployed, the value of the hyperparameters affecting the behaviour, the metrics that have been measured, etc.

Thus, each of the benchmarks provided within StairwAI is accompanied by a JSON-like file descriptor. Assuming that we have run an AI asset <asset_name>, using the HW platform <HW>, the name of the descriptor will be <asset_name>_<HW>.json.

An example of the descriptor is provided in the Appendix.

5.2.3. Data exchange Service

A data exchange service has been created to handle and struct the storage of the benchmarks and bridge the benchmarking service described in Sec. 5.2.1 and the vertical matchmaking service described in D6.5. On the one hand, the service enables any provider to upload data via an API and store the data in a structured manner. On the other hand, the vertical matchmaking component interfaces with this service, consuming the benchmark data to train its ML-based engine.

The data exchange service allows the decoupling of the services as mentioned above, as users may want to benchmark multiple algorithms (or the same algorithm on multiple HW platforms) without having to perform matchmaking. Likewise, the vertical matchmaking service gathers the datasets, which are the outcome of the benchmarking service (formatted as described by the metadata, see the Appendix), without such performing the benchmarking itself.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

The service is built as a Python Flask web service[6], and it is deployed as a Docker container. The benchmark providers can supply their data to the service through its REST APIs (POST actions). Similarly, consumers can access the data through the APIs (GET actions). More in detail, the service handles two types of files that are required by the vertical matchmaking component to use a benchmark effectively: a configuration file (JSON), as described in the Appendix, and the benchmark dataset (CSV) that follows such description. Both are expected to be provided for each algorithm run on a specific hardware platform.

In addition to the files required by the vertical matchmaking component, the data exchange service also handles a reference table with details about the hardware platforms (CSV). The table can be retrieved via a GET action and updated via a POST action (reuploading the whole file).

5.2.3.1. APIs

Next, we describe the API for the data exchange service:

Configs

- `/configs (GET)`: get list of (algorithm, hw) couples for which configs are available.
- `/configs/<algorithm>/<hw> (GET)`: retrieve the config (JSON) relative the the (algorithm, hw) couple.
- `/configs/<algorithm>/<hw> (POST)`: upload/overwrite the config (JSON) relative the the (algorithm, hw) couple.
    - Data must be passed by the file field.
- `/configs/<algorithm>/<hw> (DELETE)`: delete the config (JSON) relative the the (algorithm, hw) couple.

Datasets

- `/datasets (GET)`: get list of (algorithm, hw) couples for which datasets are available.
- `/datasets/<algorithm>/<hw> (GET)`: retrieve the dataset (CSV) relative the the (algorithm, hw) couple.
- `/dataset/<algorithm>/<hw> (POST)`: upload/overwrite the dataset (CSV) relative the the (algorithm, hw) couple.
    - Data must be passed by the file field.
- `/datasets/<algorithm>/<hw> (DELETE)`: delete the dataset (CSV) relative the the (algorithm, hw) couple.

HW info

- `/hw_info (GET)`: retrieve the CSV containing information about hardware platforms.
- `/hw_info (POST)`: upload/overwrite the CSV containing information about hardware platforms.
    - Data must be passed by the file field.
- `/hw_info (DELETE)`: remove the CSV containing information about hardware platforms.

[6] https://flask.palletsprojects.com/en/2.3.x/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

5.2.4. Alignment with AI-on-demand platform

As planned for WP6, new benchmarks have been generated by the StairwAI's partners. The benchmarks have been generated according to the service described in Sec. 5.2.1 and made available to the StairwAI platform through the data-sharing service described in Sec. 5.2.2 and 5.2.3. The key aspect of these internal benchmarks is that they were generated according to the format agreed upon with other partners of the work package to create datasets that could be fed to the vertical matchmaking component, described in deliverable 6.5. Each dataset is accompanied by a metadata descriptor, following the format reported in Appendix 7.1. This allows for seamless integration with the vertical matchmaking service.

During the course of this work package, StairwAI has also participated in an AI4Europe workshop on benchmarking. The main goal of this meeting was to align the efforts for benchmarking across multiple ICT49 projects and the AI-on-demand platform. During the workshop, multiple benchmarking frameworks and benchmark results were presented. The benchmark frameworks' alignment was less critical as they focused on different AI problems, e.g., training or inference, cloud or edge. Nonetheless, and as concluded in Section 5.1, it is fairly important to align the benchmark results on some established standard format to have a clear common understanding and smooth transition across different AI Tasks. Thus, the format provided in the Appendix for StairwAI's benchmarks is a first step towards this goal, which should be scaled and agreed upon with the broader AI-on-demand community.

To be able to use the benchmarks from the AI-on-demand platform for the vertical matchmaking engine, they need to have a sufficiently standardized format that allows users to submit a diverse set of AI assets while making them reusable. Since, to date, there is not a clear format nor matured API to use those benchmarks, it has not been possible to validate/extend the vertical matchmaking further with such datasets. Nonetheless, the effort undertaken within this work package allowed the identification of a set of guidelines for a common benchmarking strategy, with an explicit focus on the need to obtain shared metadata descriptors for the outcome of the benchmarking process. These guidelines will be shared with the AI4Europe partners and will guide the development of the general benchmarking strategy.

Further, during the rest of the StairwAI project and particularly within WP2, StairwAI's partners will push on the integration with the AI-on-Demand by sharing the internal benchmark datasets with the broader AI-on-demand community through the AI-on-demand API.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

6. Conclusion and future work

This document has presented the 2nd version of the *Benchmark results* proposed within WP6 Task 6.2 of the StairwAI project. The document has described the benchmark results obtained using StairwAI's frameworks (D6.2) across multiple heterogeneous platforms, providing a rich and heterogenous dataset (3000+ configurations) used to train the vertical matchmaking engine. Besides, we have provided a survey of external benchmarks where we analyse the characteristics and feasibility of adding external data to train the vertical matchmaking engine.

In addition, we have identified and proposed a common format to describe and standardize the output of the benchmarking results to make them reusable by the vertical matchmaking engine. Further, a data exchange service (deployed in the StairwAI platform) has been created to gather the benchmark data and bridge the benchmarking service and the vertical matchmaking service.

Finally, this document defines a clear set of APIs and guidelines to allow the integration of the benchmarks (and their reusability) in the AI-on-demand platform.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

7. Appendix

7.1. Metadata example

The following is a metadata descriptor, which can be expanded according to specific needs. This sketch contains the kernel of information that must be available to effectively run and describe a benchmark (a crucial aspect for reproducibility).

Metadata descriptor (JSON)

```
{
  "name":"algo1",
  "HW_ID":"hw1",
  "HW_price":null,
  "inputs":[
    {
      "ID":"input_0",
      "description":null,
      "type":"str",
      "LB":null,
      "UB":null
    },
    {
      "ID":"input_1",
      "description":null,
      "type":"float",
      "LB":null,
      "UB":null
    }
  ],
  "hyperparams":[
    {
      "ID":"hyperparam_0",
      "description":null,
      "type":"float",
      "LB":null,
      "UB":null
    },
    {
      "ID":"hyperparam_1",
      "description":null,
      "type":"bin",
      "LB":null,
      "UB":null
    },
    {
      "ID":"hyperparam_2",
      "description":null,
      "type":"int",
      "LB":null,
      "UB":null
    }
  ],
  "targets":[
    {
      "ID":"time",
      "description":null,
      "LB":null,
      "UB":null
    },
    {
      "ID":"memory",
      "description":null,
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

```
      "LB":null,
      "UB":null
    }
  ]
}
```

Description of the fields:

- name: name of the algorithm.
- HW_ID: name of the hardware platform.
- HW_price: default price for the hardware platform (optional).
- Structure of the inputs, hyperparameters and targets sections:
  - ID: ID of the input/hyperparameter/target.
  - type: type of the input/hyperparameter: “int”, “float”, “bin’ or “str” (look at the dataset example for more); not present for targets, which are assumed to be of type “float”.
  - description: description of the input/hyperparameter/target (to be shown to the user for guidance, optional).
  - UB and LB: upper bound and lower bound (it's extracted from the dataset if not specified).

The whole “inputs” section is optional: if present, the configuration file is intended as describing an input-dependent case; if not present, an input-independent case.

Dataset (CSV)

The dataset, following the JSON description, is expected to have at least two types of columns, hyperparameters and targets (metrics), where each row represents a measurement: the algorithm is run with the specified hyperparameters' values, resulting in the reported values for the targets.

In the input-dependent case ulterior columns are expected to represent input variables, with the values that resulted in the measurement in each row.

Dataset example (CSV) related to shown JSON:

```
input_0,input_1,hyperparam_0,hyperparam_1,hyperparam_2,time,memory
a,1.12,2.33,1,32,120.34,1022.56
b,4.23,3.55,0,74,93.78,987.85
c,7.46,2.22,1,22,44.13,1328.77
```

Columns corresponding to inputs and hyperparameters are expected to be of the type declared in the configuration file:

- integers if the type is int;
- any numerical value if the type is float;
- 0s and 1s if the type is bin;
- strings if the type is str.

Targets are assumed to be always of type float (i.e., a continuous numerical).

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 101017142