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

Benchmarking software-2nd version

Deliverable information	
Deliverable number	D6.2
WP number and title	WP6 - Vertical Matchmaking and hardware marketplace
Lead beneficiary	BCA
Dissemination level	Public
Due date	30th April 2023
Actual date of delivery	3rd May 2023
Author(s)	Miguel de Prado (BCA), Jean Marc Bonnefous (BCA), Jagyan Prasad Mahapatro (HUA), Hamdi Bouchech (HUA)

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	15/07/2022	Initial draft of document	Miguel de Prado (BCA)
0.2	22/07/2022	First version of deliverable completed	Miguel de Prado (BCA)
0.3	29/08/2022	Internal review completed	Michela Milano (UNIBO), Gabriel Gonzalez (Insight)
0.4	23/09/2022	Answers to reviewer's comments added	Miguel de Prado (BCA)
1.0	19/10/2022	Final version of the document	Diletta Rizzolo (Unibo)
1.2	31/03/2023	Extension of V1 version	Miguel de Prado (BCA)
1.4	04/04/2023	Addition of HUA SW and HW	Jagyan Mahapatro (HUA), Hamdi Bouchech (HUA)
1.7	02/05/2023	Revision by QA	Michela Milano (UNIBO), Gabriel Gonzalez (Insight)
2.0	03/05/2023	Final version	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

Acronyms

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

In M20, the 1st version of the *Benchmark software framework* was released within WP6 Task 6.1 of StairwAI project. The 1st version of *benchmark software framework* represented one of the main blocks of the vertical matchmaking service as it produced the benchmarks across several HW platforms (Task 6.2) that are then used to train the vertical matchmaking engine (Task 6.3).

This document provides the 2nd version of the *Benchmark software framework,* including the integration of a new SW inference engine and HW platform from Huawei into LPDNN. Besides, this deliverable also provides a large step towards achieving MS10 for the integration of the Benchmark as a Service into the AI-on-demand platform.

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

Section 2 provides a general introduction of the WP and summarize the scope and contributions of this deliverable.

Section 3 introduces LPDNN: the inference framework that has been used and extended to create a benchmark framework for heterogeneous HW platforms. First, LPDNN's architecture is detailed, explaining the different components that make it an interoperable framework. Next, we introduce the different inference engines that LPDNN supports, which increase the portability and optimisation of neural networks across HW platforms. Then, we describe the support for HW platforms and those platforms that have been already integrated.

Section 4 starts by explaining the elements that are used to deploy AI Applications on HW platforms. Then, the benchmark flow is explained by showcasing the execution of an AI Application and the obtention of a benchmark.

Section 5 introduces the design of the Benchmark as a Service (WP2) out of the Benchmarking software framework, increasing its scope and usability as well as its integration into the AI-on-demand platform.

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

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.

As such, one of the main objectives of WP6 is to develop *benchmark software framework/service* suitable for selected machine learning models and hardware, including CPU (Intel x86, Arm Cortex-A5x, Risc-V), GPGPU (NVIDIA, etc), and NPU (HUA, etc) accelerated platforms. The benchmark software framework will be used in Task 6.2 to produce the benchmarks and create a profiling dataset that can be used in Task 6.3 to train the Vertical Matchmaking engine's algorithms.

In this document, we propose LPDNN framework as the *benchmark software framework.* LPDNN framework was developed during the H2020 Bonseyes project (Prado, Miguel De, et al) and has been largely extended to provide a structed benchmarking layer to analyze the execution of AI Applications, particularly DNNs, 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 (standarization)
  - Easy to replicate (reproducibility)
  - Create trust & community

Taking the previous design principles, the main contributions of the *benchmark software framework* are the following:

- Introduction of LPDNN as an inference framework.
- Integration into LPDNN of four inference engines to benchmark CPU, GPU and NPU platforms.
- Integration into LPDNN of four HW platforms and update of another four available platforms to the latest SW release.
- Creation of the benchmarking layer within LPDNN, containing a variety of static and dynamic metrics.
- Design of the benchmarking framework as Service that will be used by external users and interoperable with the AI-on-demand platform.

2.1. Purpose and Scope of the document

Deliverable D6.2 is a Demonstrator, i.e., it introduces the second and final version of the *Benchmark software framework.*

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

3. LPDNN

LPDNN, which stands for Low-Power Deep Neural Network, is a deployment framework that provides the tools and capabilities to generate portable and efficient implementations of DNNs. The main goal of LPDNN is to provide a set of AI applications for deep learning tasks, e.g., object detection, image classification, speech recognition, which can be deployed and optimised across heterogeneous platforms, e.g., CPU, GPU, FPGA, NPU.

Figure 1 LPDNN AI classes

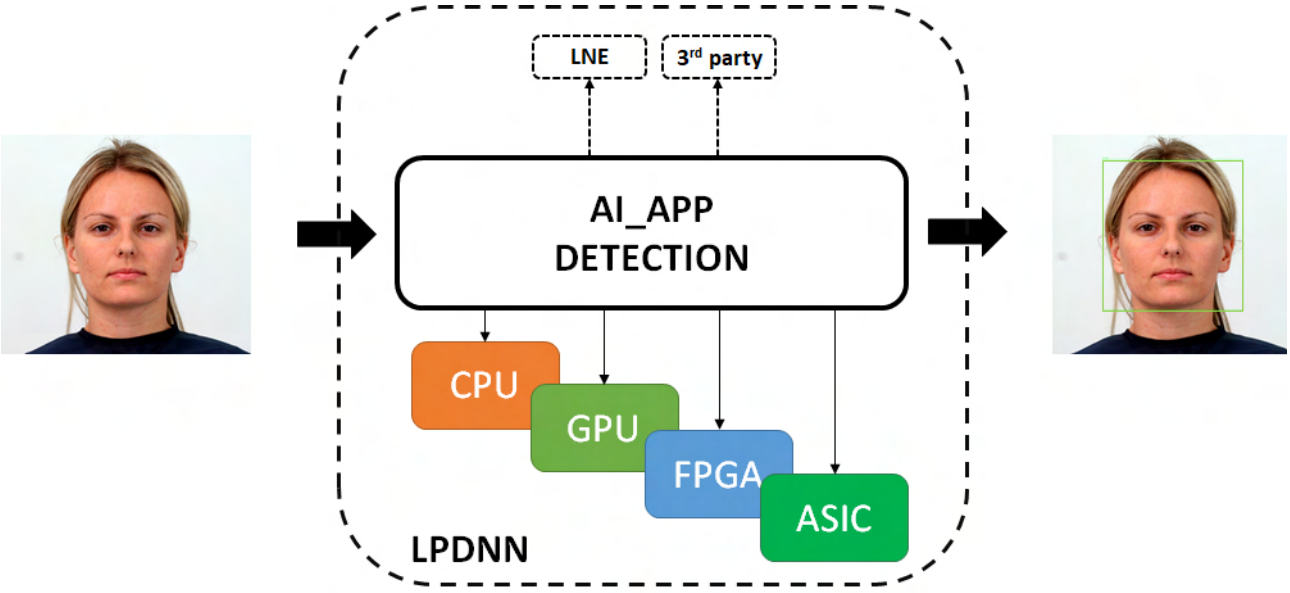

Figure 2 LPDNN overview

3.1. LPDNN architecture

One of the main issues of deep learning systems is the hardship to replicate results across different systems. To solve this issue, LPDNN features a full development flow for deep learning solutions on embedded 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 across systems.

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

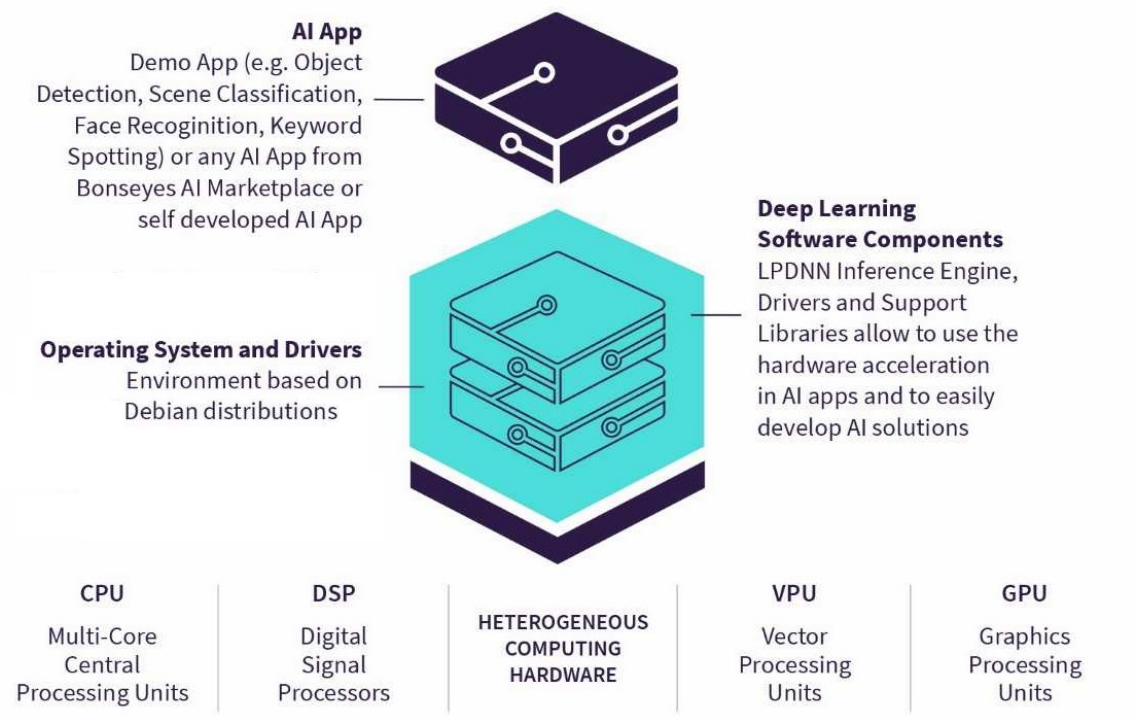

Figure 3 LPDNN full stack

AI applications are the result of LPDNN's optimisation process and the higher level of abstraction for the deployment of DNNs on a target platform. They contain all the necessary elements or modules for the execution of a DNN. An AI application is the optimised representation of a DNN model to be efficiently executed on a target embedded device. An AI application contains all the necessary elements or modules for the execution of a DNN:

- **Pre-processing:** Step to prepare, normalize or convert the input data into the required input that is expected by the DNN.
- **DNN inference:** Forward-pass of the neural network. The execution is taken care of by an inference engine.
- **Post-processing:** Conversion of the neural network's output into structured and human-readable information.

Next, we detail LPDNN's architecture by further describing the concept of LPDNN 's inference engines and the support for heterogeneous platforms.

3.2. Inference engines

AI applications contain a hierarchical but flexible architecture that allows new modules to be integrated within the LPDNN framework through an extendable and straightforward API. For instance, LPDNN supports the integration of 3rd-party self-contained inference engines to perform DNN inference. Initially, LPDNN only supported:

- **LNE**: LPDNN Native Engine (LNE) allows the execution of DNNs across arm-based and x86 CPUs as well as on Nvidia-based GPUs.

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

During the development of the 1st version of the *Benchmark software framework* in the Stairwai project, the following inference engines have been integrated into LPDNN:

- **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.

The inclusion of external engines also benefits LPDNN as certain embedded platforms provide their own specific and optimised framework to deploy DNNs on their hardware.

As part of the 2nd release of the benchmarking software framework, Huawei's inference engine has also been integrated. Huawei's computing hardware fall into the category of platforms with a dedicated inference engine, called Ascend Computing Language (ACL). ACL provides a collection of C language APIs for users to develop deep neural network apps ranging from device management and context management to model/operator loading and execution.

ACL APIs can be called through a third-party framework such as LPDNN to utilize the compute capability of the Ascend AI Processor. The integration of ACL into LPDNN provides a method for benchmarking AI apps on Huawei's platforms against other platforms such as GPUs and CPUs and other inference engines such as NCNN, ONNXRuntime, etc. with a common framework. Further information can be found here.

More information about how to add a new inference engine in LPDNN can be found at:

- Developer guides: https://bonseyes.gitlab.io/bonseyes-cli/pages/dev_guides/ai_app_index.html

3.3. Supported HW platforms

One of the main factors for LPDNN's adoption is performance **portability across the wide span of hardware platforms**. LPDNN's flexible architecture allows the main core to remain small and dependency-free while additional 3rd party libraries or inference engines are only included when needed and for specific platforms, notably increasing the portability across systems. Besides, cross-compilation and specific tools are added to support a **wide range of heterogeneous computing platforms such as CPUs, GPUs, NPUs.** One of the objectives of LPDNN is to provide full support for reference platforms by providing:

- **Developer Platform Environments (DPEs)** to help the user employ a developer platform, including OS images, drivers, and cross-compilation toolchains for several heterogeneous platforms.
- **A dockerised and stable environment**, which increases the reliability by encouraging the replication of results across platforms and environments.
- **Optimisation tools and computing libraries** for a variety of computing embedded platforms that can be used by LPDNN's inference engines to accelerate the execution of neural networks.

Originally, the range of embedded platforms that were supported within LPDNN were the following:

- **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

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

- **Nvidia Jetson Xavier:** Octa-core ARM v8.2 @ 2.03 GHz & 512-core Nvidia Volta GPU with Tensor Cores

During the development of the 1st version of the *Benchmark software framework* in the Stairwai project, we have employed Bonseyes LPDNN's platform workflows to integrate new HW platforms and provide a more heterogeneous set of benchmarks on variety of HW processing cores. The following HW platforms have been fully integrated:

- **Intel NUC:** Intel quad-core (TM) i5-7260U CPU @ 3.4 GHz
- **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

Besides, the following platforms have been upgraded:

- **Nvidia Jetson Nano:** Update to latest jetpack release JP6.4
- **Nvidia Jetson Xavier:** Update to latest jetpack release JP6.4
- **Raspberry Pi 3b+:** Update to latest Ubuntu20 packages
- **Raspberry Pi 4b:** Update to latest Ubuntu20 packages

As part of the 2nd release of the benchmarking software framework, two new platforms from Huawei have also been used for deployment:

- **Atlas 200 DK (model: 3000):** The Atlas 200 DK is a high-performance AI application developer board that integrates the Ascend 310 AI processor to facilitate quick development and verification. It has been widely used in scenarios such as developer solution verification, higher education, and scientific research.
- **Atlas 800 Training Server (Model: 9010):** The Atlas 800 training server is an 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.

Further information can be found here.

Furthermore, BonsAPPs project is currently performing the integration of extra-low-power platforms containing Micro-controller Units, which can then be leveraged in Stairwai for benchmarking:

- **STM32 H7A3**: Arm® Cortex®-M7 core @ 480 MHz & 1.4 Mbytes of SRAM
- **Greenwaves GAP8**: RISC-V core & Octa-core RISC-V (cluster) & 512kB of L2 memory

More information about the HW platforms can be found at:

- User guide: https://bonseyes.gitlab.io/bonseyes-cli/pages/user_guides.html#platform
- Developer guide: https://bonseyes.gitlab.io/bonseyes-cli/pages/developer_guides.html#platform

Developer platforms can be accessed upon request at https://gitlab.com/bonseyes/platforms/.

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

3.4. Benchmark framework

LPDNN provides a structed benchmarking layer to analyze the execution of AI Applications on HW platforms. LPDNN integrates a first analysis of static metrics during the offline compilation of the AI Applications. Metrics such as FLOPS, model parameters and model storage are obtained. Besides, LPDNN integrates two methods to benchmark the execution of AI Apps:

- **Built-in anchors:** C++ anchors are included in LPDNN to measure the latency and memory consumption of the AI Apps during their executions. Several anchors are included:
  - Pre-processing: This anchor measures the latency and memory of a given pre-processor:
    - Image: cropping, normalization, resize, filtering
    - Audio: mfcc feature computation
    - Signal: resizing, filtering
  - Inference: This anchor measures the latency and memory spent during the forward pass of the neural network for a given inference engine.
  - Post-processing: This anchor measures the latency and memory spent during the post-processing of the outputs of the forward pass, e.g., decoding, non-max suppression, bounding box forming.
  - Total execution: This anchor measures the latency and memory spent during the total execution of the AI App.
- **HW resources monitoring:** Python script that monitors the HW resources of the system while the execution of the AI App. Metrics such as CPU/GPU/NPU processor load, peak/average memory usage, memory bandwidth, temperature, etc.

Overall, these are the following metrics are LPDNN collects:

- **Static metrics (not measured on device):**
  - FLOPs
  - Parameters
  - Storage
- **Dynamic metrics (measured on device):**
  - Latency
  - Throughput
  - Average Memory (CPU/GPU)
  - Peak Memory (CPU/GPU)
  - Memory Bandwidth (CPU/GPU)
  - Processor load (CPU/GPU/NPU)
  - Power consumption
  - Temperature

For more details, please refer to D6.3 submitted in M25.

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

4. Deployment of AI Apps and Benchmark

In this section, we showcase what elements are needed for deployment and how to execute a benchmark.

4.1. Deployment of AI Apps

To be able to execute AI applications on a hardware platform, two elements are required:

A. An LPDNN AI application:

It defines the class and structure of the AI application, the DNN models' architecture and weights, its deployment configuration and the pre- and post-processing that it takes. LPDNN AI applications are platform-specific, although the same AI application can be executed on different HW platforms as long as the selected inference engine or backends are supported by the HW platform.

B. An LPDNN Deployment Package:

Collection of tools, executables, libraries, inference engines and backends that allows the actual execution of the LPDNN AI application. The collection of libraries and binaries that are copied on the target platform for the execution of the DNN is called a *runtime*. The runtime dynamically loads an AI application and executes it based on its defined configuration. LPDNN's deployment packages are platform specific as they contain the inferences engines and backends supported by the HW platform.

4.1.1 LPDNN AI Application (AI App)

An LPDNN AI App is composed of the following files:

- **ai_app_config.json**: This file is the main descriptor of an AI App. It defines the AI App's components and their type, e.g., image_classification, object_clasification, face_recognition, audio_classification andsignal_processing, the type of pre- and post-processing as well as the inference engine to use to execute the DNN model. It also points to the model architecture and weights file.
- **ai_app.yml**: This file defines the AI App metadata and license type. It also This describes the platform, runtime and challenge that the AI App was initially compiled for. This file is not used by the runtime, but by other deployment tools.
- **DNN model**: A DNN model describes the model architecture and the trained weights. A DNN model may come on different forms based on the selected inference engine, e.g., model.json, model.param, model.bin or model.onnx.

4.1.2 LPDNN Deployment Package

An LPDNN runtime, contained within a deployment package, is the collection of binaries and libraries that are copied to the target platform for the execution of DNNs. Runtimes are composed of the following files:

- **Binaries**: This folder contains the set of executables to start an AI App manually through different interfaces, e.g., ai-app-cli, ai-app-cli.py, http-worker.
- **Libraries**: This folder contains the set of dynamic and static libraries that are included in LPDNN for a target developer platform. It includes the inference engines, the pre- and post-processing routines, backends, etc.
- **Solutions**: This folder contains the bash scripts to start up an AI App automatically or remotely.
- **package.yml**: This file defines the runtime name.

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

- ❖ **runtime.yml**: This file contains metadata for the runtime and it is used to automatically start an AI App using the Bonseyes tools.
- ❖ **engines.yml**: This file describes the available engines for within the runtime.

4.2. Benchmark of AI Apps

Fig. 4 shows the Bonseyes Benchmark User Journey to benchmark an AI Application on a computing platform.

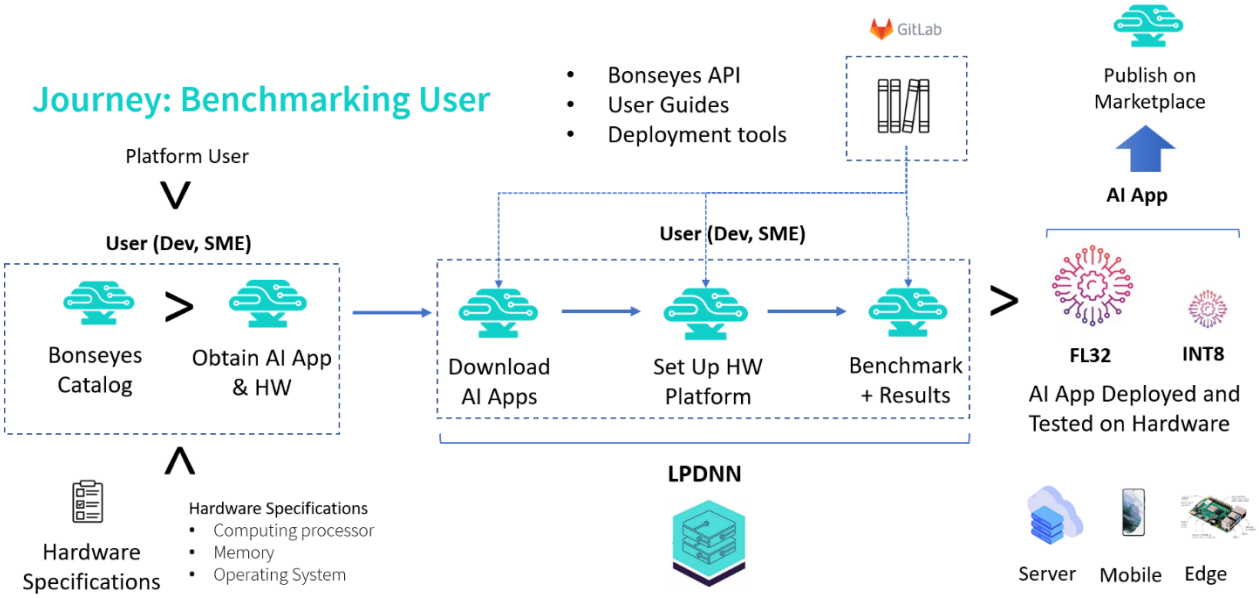

Figure 4 Benchmark User Journey

More details are given in Fig.5, which illustrates the and process a user needs to go through to benchmark an AI Application. The process is divided into the following steps:

1. Choose AI App and target HW platform from the Catalog.
2. Follow the requirements step to be compliant with the process and the tools.
3. Download an AI App and a deployment package.
4. Benchmark your AI App on your target HW.

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

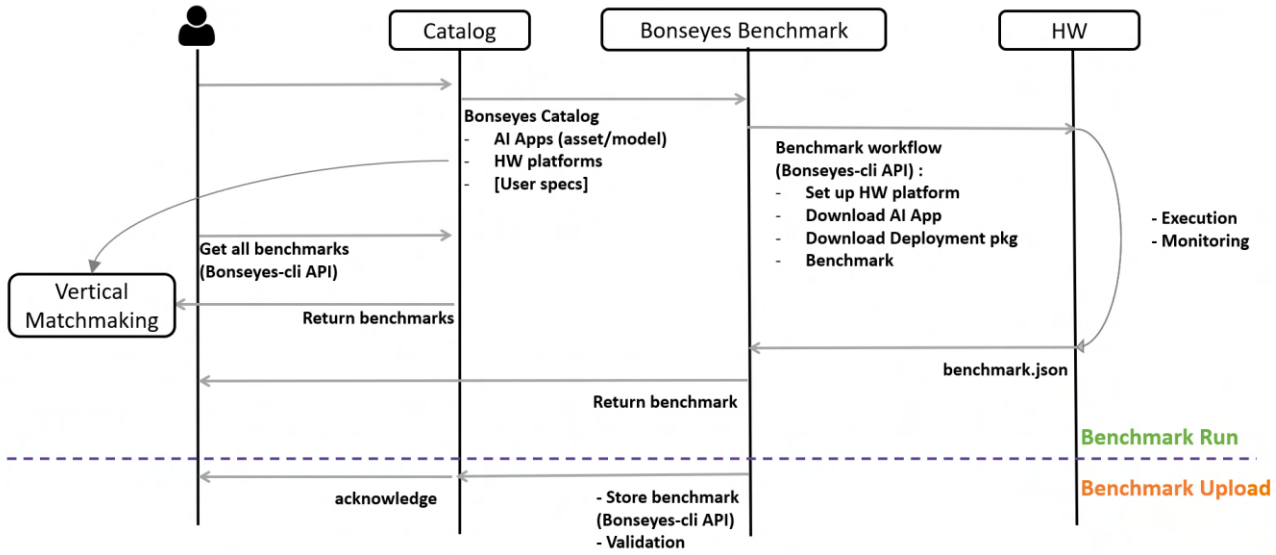

Figure 5 Benchmark procedure

Next, each of these steps will be explained in detail.

4.2.1. Choose AI App and target HW platform

The user needs to choose a target platform and AI App from the Marketplace's Catalog.

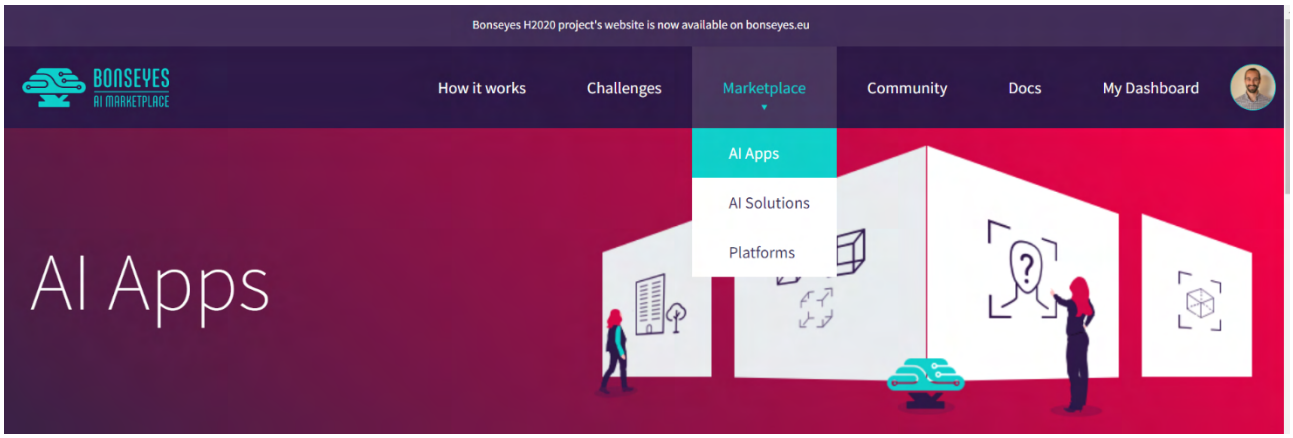

Figure 6 Bonseyes Marketplace Catalog

If the user would like to create its own application, detailed documentation is provided to generate a new one at Bonseyes docs for AI-App generation.

4.2.2. Requirements

To be able to execute AI applications on a hardware platform, the following steps need to have been performed:

1. Setup the local environment as explained in Bonseyes Prerequisites doc.
2. Set up the target hardware as explained in Setup platform section of Bonseyes' doc and have the *${platformName_src} ${platformName_build} and ${platformName_config}* folders in your machine.
3. Install python packages in the target board as explained in the Packages's section.

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

Once those steps are completed, change directory to the folder where you built your target platform during the Setup platform section, e.g., my-bonseyes-platform.

4.2.3. Download AI App and deployment package

New AI App can be generated by following the Bonseyes docs for AI-App generation. Already built AI Apps can also be obtained from the Bonseyes Marketplace's catalogue. To download an AI-App from the Marketplace's catalogue execute the following command (replace ${ai_app} by the name you would like to give to your AI-App):

```
bonseyes marketplace download-ai-app --output-dir ${ai_app}
```

A dialog will prompts asking you to choose the AI-App you would like to download and he ai app will be downloaded in the directory `${ai_app}` as shown in Figure 7. The AI App will contain a series of files, as explained in Section 4.1.1.

```
→ ~ bonseyes marketplace download-ai-app --output-dir ai_app
Please choose ai app:

1 ) Face Detection Retinaface mobile0.25mod4lite-32-72-320 for Jetson Xavier Jetpack4.6
2 ) Hello World

Please select a choice > 1
100%|                                                                                | 6.33M/6.33M [00:00<00:00, 9.39MiB/s]
Download finished
Extracting files...
./README.md
./ai_app.yml
./ai_app_config.json
./evaluation_report.yml
./lpdnn_netspec.dot
./model.bin
./model.json
./model.onnx
Done.
Removing archive... Done.
100%|                                                                                | 859/859 [00:00<00:00, 546kiB/s]
Download finished
Extracting files...
16_user_licence/user_license.json
16_user_licence/user_license.json.sha256
Done.
Removing archive... Done.
```

Figure 7 AI-App download

LPDNN's deployment packages are currently available at https://gitlab.com/bonseyes/artifacts/deployment-packages/-/packages and can be accessed under request. Once you obtained a platform-specific deployment package, copy your LPDNN's deployment package in the same folder as the target platform and the AI-App. Next, decompress it (change *${deployment_package}* by the name of your package):

```
cp ${deployment_package}.tar.gz my-bonseyes-platform
tar -xvf ${deployment_package}.tar.gz
```

4.2.4. Benchmark your AI-App on your target HW

To benchmark an AI-App, your folder should contain the following elements:

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

1. **${platformName_config}:** target configuration coming from the DPE setup
2. **${deployment_package}:** downloaded from Gitlab
3. **${ai_app}:** downloaded from the BMP

To benchmark an AI App, you need to execute the following command:

```
bonseyes ai-app benchmark-analyzer --target-config ${platformName_config} \
                                   --ai-app ${ai_app} \
                                   --deployment-package ${deployment_package} \
                                   --dataset PATH/TO/DATASET_FOLDER \
                                   [--number 20] \
                                   [--filename ${FileName}]
```

For instance, to benchmark a Face Detection Retinaface AI-App for Imagenet on the Jetson Xavier platform looks like:

```
        bonseyes ai-app benchmark-analyzer
                        --target-config jetson_xav_config \
                        --ai-app my-aiapp \
                        --deployment-package xav_deployment_package \
                        --dataset ../samples \
                        --number 10 \
                        [--filename benchmark.json]
```

This call will perform then inferences for Face Detection Retinaface AI-App on the Jetson Xavier platform with Jetpack 4.6 as follows:

```
→ jetson_xavier bonseyes ai-app benchmark-analyzer --target-config xav_config\
                            --ai-app my-aiapp-trt \
                            --deployment-package xav_deployment_package \
                            --dataset ../samples/ \
                            --number 10
Manager has been started http://0.0.0.0:49411
Success connecting at iteration 1458
-- Running inference.....
Number of objects found:  2
        class:  1 ( 0.9983725547790527 )
        class:  1 ( 0.903798520565033 )
Number of objects found:  1
        class:  1 ( 0.9978792667388916 )
Number of objects found:  1
        class:  1 ( 0.996670126914978 )
Number of objects found:  1
        class:  1 ( 0.9991635084152222 )
Number of objects found:  2
        class:  1 ( 0.9983725547790527 )
        class:  1 ( 0.903798520565033 )
Number of objects found:  1
        class:  1 ( 0.9978792667388916 )
Number of objects found:  1
        class:  1 ( 0.996670126914978 )
Number of objects found:  1
        class:  1 ( 0.9991635084152222 )
Number of objects found:  2
        class:  1 ( 0.9983725547790527 )
        class:  1 ( 0.903798520565033 )
Number of objects found:  1
        class:  1 ( 0.9978792667388916 )
['CPU_LOAD', 'CPU_MEMORY', 'CPU_MEMORY_PEAK', 'CPU_TEMPERATURE', 'GPU_LOAD', 'GPU_MEMORY', 'GPU_MEMORY_PEAK', 'GPU_TEMPERATURE', 'NPU_LOAD', 'MEMORY_BW', 'MEMORY_BW_PEAK', 'POWER']
['82.0', '10.3', '3.617481231689453', '39.5', '0.0', '3.7125092537458784', '1.1575698852539062', '36.5', '0', '0', '1', '0']
['BENCHMARK_STATE', 'OK']
Inference           :   Calls       Real            CPU          Real STD          CPU STD       Real VAR          CPU VAR    Data Type    Plugin
-------------------------------------------------------------------------------------------------------------------------------------------------------
image_preprocessor          :     9     8.649111111     8.649666667     2.452978074     2.453223186     6.017101432     6.018304000
engines/tensorrt            :     9     6.866666667     6.867000000     0.333072898     0.332955452     0.110937556     0.110859333
image_postprocessor         :     9     4.865444444     4.865222222     0.368915862     0.368970272     0.136098914     0.136139062
object_detection            :     9    20.381222222    20.381888889     2.416225099     2.416095861     5.838143728     5.837519210
-------------------------------------------------------------------------------------------------------------------------------------------------------
```

Figure 8 Benchmark with LPDNN

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

To store the benchmark in a file, add the *–filename* option and the metrics will be dumped in the named file in JSON format. If the file name option is enabled, prediction results will be automatically dump into *results.txt**.

5. Benchmark as a Service

As part of the 2nd release of the benchmarking software framework, we also introduce the Benchmark as-a-Service (BaaS) design in collaboration with WP2 for broader interoperability with the AI-on-demand platform. StairwAI aims to enhance the AI-on-demand platform services by employing a service layer that provides, among others, Vertical Matchmaking. The Benchmarker, a building block of the Vertical Matchmaking, was not initially considered as a service on its own. Nonetheless, given its maturity, the StairwAI has provided the means and effort for a closer integration of the BaaS into the AI-on-demand platform.

Partners of WP6 have held coordination meetings with AI4Europe partners to understand the requirements and dependencies required to interoperate the BaaS into the AI-on-demand platform. Given that the AI-on-demand platform may contain multiple AI artifacts, e.g., dataset, AI model, or algorithm, the best solution found was to provide interfaces that define the API to communicate or transfer the content from one service/database to another. Thus, AI models from the AI-on-demand platform could be benchmarked through the BaaS from StairwAI. Fig. 9 illustrates the interfaces between the AI-on-demand Platform, the AI artifacts from AI4EU and the BaaS:

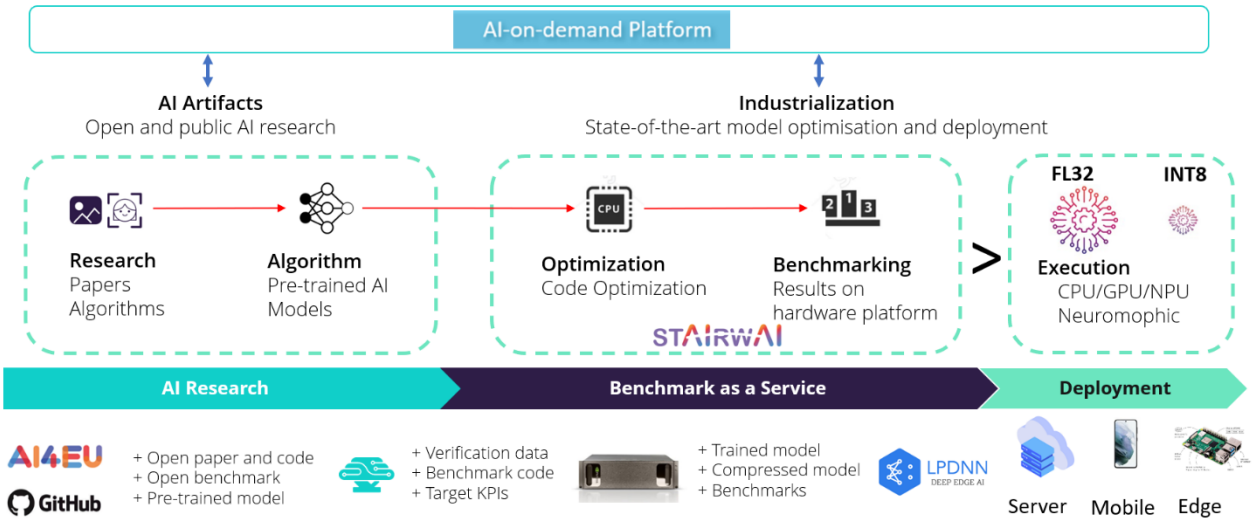

Figure 9 Benchmark as a Service workflow

5.1. BaaS Design

To be able to put in place such a workflow, it was envisioned a service layer wrapping and also extending the *Benchmarking Software framework* (middle green block in Fig.9). This would also allow to alleviate some of the *Benchmarking software framework* dependencies and be able to benchmark any AI model and format while still benefiting from LPDNN optimisations. The dockerized service that is proposed is described in Fig. 10 and will allow the following functions:

- Transfer of AI models from the AI-on-demand Platform to the BaaS for benchmarking by providing CLI or HTTP interfaces.

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

- Dispatch of requests from the central service to the operative docker containers to benchmark the AI model on the target platform.
- Check the format and compatibility of the AI model with the available inference engine on the target platform when the model is received at the docker container. Note that some computing platforms do not support all inference engines, e.g. RPI does not support TensorRT as it does not have an Nvidia GPU.
- Benchmark of the AI model on the target platform and return of the results to the end user through the dispatcher.

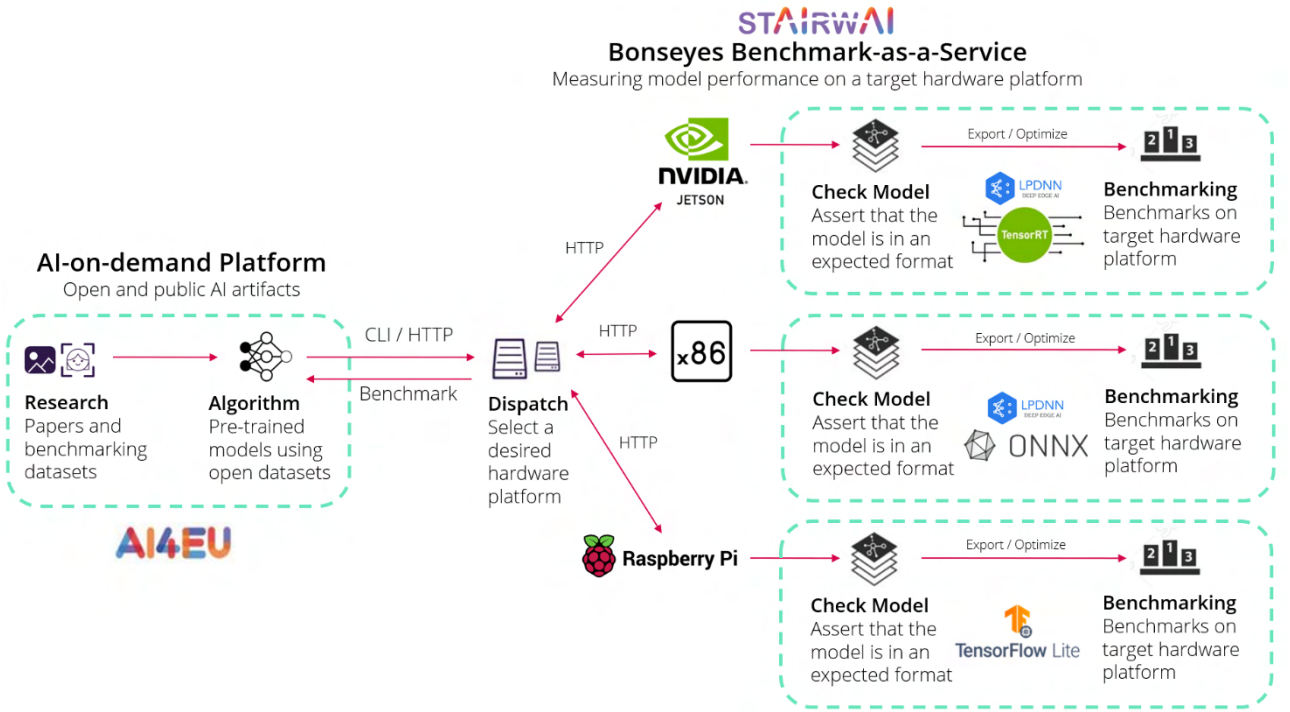

Figure 10 Benchmark as a Service implementation

In order to transform the *Benchmarking software framework* into the BaaS described in Fig. 10, LPDNN will be released in the form of deployment packages that can be employed within the B*aaS* workflow for AI model inference. LPDNN deployment packages will fit into a more abstract and general class for AI model inference, which also allows to add not supported engines such as TensorFlow Lite and broaden the range of AI model that can be benchmarked. LPDNN benchmarking layer, explained in Section 3.4, will be adapted to provide all the available metrics, i.e., static, and dynamic, in a coherent manner for all inference engines. Finally, to create the BaaS's docker containers, LPDNN's DPE, explained in Section 3.3, will be used to include all the environment, libraries, and dependencies to set up and deploy AI models on the platform.

More details and efforts will be put into the integration and deployment BaaS with the AI-on-demand platform as part of T2.5 during the last months of the StairwAI project.

5.2. BaaS API

In this section, we introduce the API that will be available for the BaaS once it's deployed. We provide two types of interfaces:

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

5.2.1. CLI

Command Line Interface (CLI) allows to a given user to query the BaaS locally through the command line, allowing for easy usage and testing. The usage of the BaaS would be as follows:

```
Usage: bonseyes_baas_cli
       --model-path <local_model_path>
       --engine {tf, tflite, onnx, tensorrt, lpdnn}
       --profile (specifies versions of dependencies etc.)
       --platform {x86, raspberrypi, jetson}
       [--device {cpu, gpu}]
```

The Bonseyes_baas_cli will connect to the dispatcher, which based on the platform and profile, will redirect the query to the right docker container. The docker container will then take the model-path and engine to perform the benchmark either on the CPU or GPU as specified by the *device* parameter.

5.2.2. HTTP

The HTTP interface allows to a given user to query the BaaS remotely without direct access to where the BaaS is deployed (only URL needed). The usage of the BaaS would be as follows:

```
<url>/bonseyes_baas/<platform>/<profile>/<engine>?device=<cpu|gpu>
       <platform>: {x86, raspberrypi, jetson}
       <engine>: {tf, tflite, onnx, tensorrt, lpdnn}
```

For a user to be able to post a query, the model should be in a binary body of the request as follows:

```
curl --request POST --data-binary @<path_to_model_file>
<url>/bonseyes_baas/<platform>/<profile>/<engine>?device=<cpu|gpu>
```

The BaaS internal process will perform in the exact same way as explained for CLI.

6. Conclusion and future work

This document has presented the second and final release of the *Benchmark software framework* proposed within WP6 Task 6.1 of StairwAI project. The document describes how LPDNN inference framework has been used and extended to accommodate a benchmark layer that allows to profile AI Applications on a range of heterogeneous HW platforms. The deployment and benchmark of AI Applications has been showcased, illustrating the flow and commands that a user need to execute to benchmark an AI Application.

In addition, we have presented the design of the Benchmark as-a-Service, extending the Benchmark software framework's scope to increase its reusability and interoperability with the AI-on-demand platform and allow the benchmark of AI-on-demand's assets. This represents a large step towards achieving MS10 for integrating the services into 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

Bibliography

1. Prado, Miguel De, et al. "Bonseyes AI pipeline—Bringing AI to you: End-to-end integration of data, algorithms, and deployment tools." *ACM Transactions on Internet of Things* 1.4 (2020): 1-25.

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