Ref. Ares(2019)4132677 - 29/06/2019

software framework for runtime-Adaptive and secure deep Learning On Heterogeneous Architectures

Project Number 780788

Project Acronym ALOHA

D1.2	Report on tool flow integration		
Work Package:	WP1	**Lead Beneficiary:**	UNICA
Type:	Report	**Dissemination level:**	Public
Due Date:	30/06/2019	**Delivery:**	28/06/2019
Version:	1.0		

Brief description:

This deliverable reports on the integration activities carried out by the Consortium in Task 1.3 from month M7 to M18 (July 2018 – June 2019) for integrating the different components and tools developed in work packages WP2, WP3 and WP4 into an integrated functioning framework.

Deliverable Author(s):

Name	Beneficiary
Paolo Meloni	UNICA
Daniela Loi	UNICA

Deliverable Revision History:

Reviewer Beneficiary	Issue Date	Version	Comments
UNICA	06/05/2019	V0.1	Released of the Table of Content and introduction
UNICA	31/05/2019	V0.2	Draft release
UNICA	03/06/2019	V0.3	Added section 4
PLURIBUS	20/06/2019	V0.4	Added KPI2_2 and KPI2_3
UNICA	24/06/2019	V0.5	Integrations on section 4
UNICA	26/06/2019	V0.6	Final draft
UPF	27/06/2019	V0.7	Review
UNICA	28/06/2019	V1.0	Final version including reviewer's suggestions

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

Disclaimer

This document may contain material that is copyright of certain ALOHA beneficiaries, and may not be reproduced, copied, or modified in whole or in part for any purpose without written permission from the ALOHA Consortium. The commercial use of any information contained in this document may require a license from the proprietor of that information. The information in this document is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

The ALOHA Consortium is the following:

#	Participant Legal Name	Acronym	Country
1	STMICROELECTRONICS SRL	ST-I	Italy
2	UNIVERSITA' DEGLI STUDI DI CAGLIARI	UNICA	Italy
3	UNIVERSITEIT VAN AMSTERDAM	UVA	Netherlands
4	UNIVERSITEIT LEIDEN	UL	Netherlands
5	EIDGENOESSISCHE TECHNISCHE HOCHSCHULE ZUERICH	ETHZ	Switzerland
6	UNIVERSITA' DEGLI STUDI DI SASSARI	UNISS	Italy
7	PKE HOLDING AG	PKE	Austria
8[1]	**CA TECHNOLOGIES DEVELOPMENT SPAIN SA**	**CA**	**Spain**
9	SOFTWARE COMPETENCE CENTER HAGENBERG GMBH	SCCH	Austria
10	SANTER REPLY SPA	REPLY	Italy
11	IBM ISRAEL - SCIENCE AND TECHNOLOGY LTD	IBM	Israel
12	SYSTHMATA YPOLOGISTIKIS ORASHS IRIDA LABS AE	IL	Greece
13	PLURIBUS ONE SRL	PLURIBUS	Italy
14	MAXQ ARTIFICIAL INTELLIGENCE, LTD	MaxQ-AI	Israel
15[2]	**UNIVERSIDAD POMPEU FABRA**	**UPF**	**Spain**

[1] The participation of CA TECHNOLOGIES DEVELOPMENT SPAIN SA (CA) has been terminated on May 9th, 2019.
[2] UNIVERSIDAD POMPEU FABRA (UPF) has entered into force as new beneficiary on May 29th, 2019.

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

Table of Contents

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

Figures

Tables

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

1 Executive Summary

ALOHA is a tool flow composed by a set of modules that has been created with the purpose of facilitating the design of Deep Learning (DL) applications and their porting on embedded heterogenous architectures, by making this process as simple and painless as possible.

The main problem with the current DL solutions development flow is that the training phase leads to the selection of an optimal algorithm configuration mainly considering accuracy as the only main design objective. The selected algorithm configuration has little or no correspondence with the specific features of the processing hardware architecture in charge of executing the inference task. Developers of hardware-software systems dealing with the inference process commonly start from pre-trained networks, trying to optimize as much as possible their execution on the target computing platform. This dichotomy determines the need for multiple design iterations, potentially leading to long tuning phases, overloading designers and with results highly depending on their skills.

The ALOHA tool flow aims at automating different design steps and reducing development costs and time, by bridging the gap between DL algorithm training and inference phases. It considers hardware-related variables and security, power efficiency, and adaptivity aspects during the whole development process, from pre-training hyperparameter optimization and algorithm configuration to deployment.

To achieve the stated objectives, in Task 1.1 (*System specification and subsystem definition*) the Consortium has defined the architecture of the ALOHA toolflow, specifying its different building blocks and the interfaces among them, in compliance with requirements and specifications established by the developing partners and the use-case providers in Task 1.2 (*Use-cases specification*). For further information about toolflow components working principle and collected requirements, please refer to deliverables D1.1 (Report on general specifications and interface definition), D1.4 (General specification of the Surveillance use-case), D1.5 (General specification of the Smart-Industry use-case), and D1.6 (General specification of the medical application use-case).

This document is related to the activities carried out in work package WP1, within **Task 1.3 – ALOHA Tool flow integration**. It specifies the activities performed by the Consortium from month M7 to M18 (July 2018 – June 2019) for integrating the different components and tools under development in work packages WP2, WP3 and WP4 into an integrated functioning toolflow. This deliverable provides a generic view on the integration process, pointing out the different steps that have led to a first release of the ALOHA toolflow, which was used by use-case providers in WP5 to build their demonstrators (see deliverable D5.6) and to evaluate the functionalities and usability of standalone components. All the integration process is based on the specification activities previously carried out in Task 1.1 and Task 1.2.

The ALOHA project follows an Agile software development approach, thus the integration steps described in this deliverable will be performed iteratively, improving the first version of the ALOHA toolflow step by step and increasing its maturity continuously. The interactions among different components of the architecture are subject to possible modifications in future iterations. This document will evolve and gain more precision and substance during the next months. An updated version will be submitted at month M30:

- D1.3 Report on tool flow integration – Update [M30]

After an overview of the ALOHA framework (Section 2), its steps (Section 3), and a description of the developed user interface (Section 4), we provide in Section 5 details on the integration process and in Section 6 an overview of the testing activities performed so far. We conclude this report reporting in Section 7 an updated list of the ALOHA key performance indicators.

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

All the ALOHA partners have contributed to this deliverable.

1.1 Acronyms and abbreviations

Acronym	**Meaning**
AOW	Architecture Optimization Workbench
API	Application Programming Interface
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Networks
DSE	Design Space Exploration
GA	Genetic Algorithm
KPI	Key Perfomance Indicator
KPN	Kahn Process Network
M	Month
ONNX	Open Neural Network Exchange Format
PI	Parsimonious Inference
REST	Representational State Transfer
SDF	Synchronous DataFlow

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

2 ALOHA framework

This section describes the ALOHA framework and provides a generic view on how the different parts interact with one another to achieve the requested functionalities and performance. The purpose of the framework is to automate and facilitate three different steps: algorithm selection (STEP 1), application partitioning and mapping (STEP 2), and deployment on target hardware (STEP 3).

The overall ALOHA framework is shown in Figure 1. Through the user interface, the ALOHA tool flow receives as inputs the following data:

- a starting neural network,
- a dataset,
- a target architecture description,
- a configuration file containing information about the target application,
- a set of constraints on accuracy, security, performance and power that the application must satisfy.

Starting from this set of user-specified inputs, the tool flow generates step by step a partitioned and mapped neural network configuration, ready to be ported on the target processing architecture, that co-optimizes both the application-level accuracy and the required security level, inference execution time and power consumption.

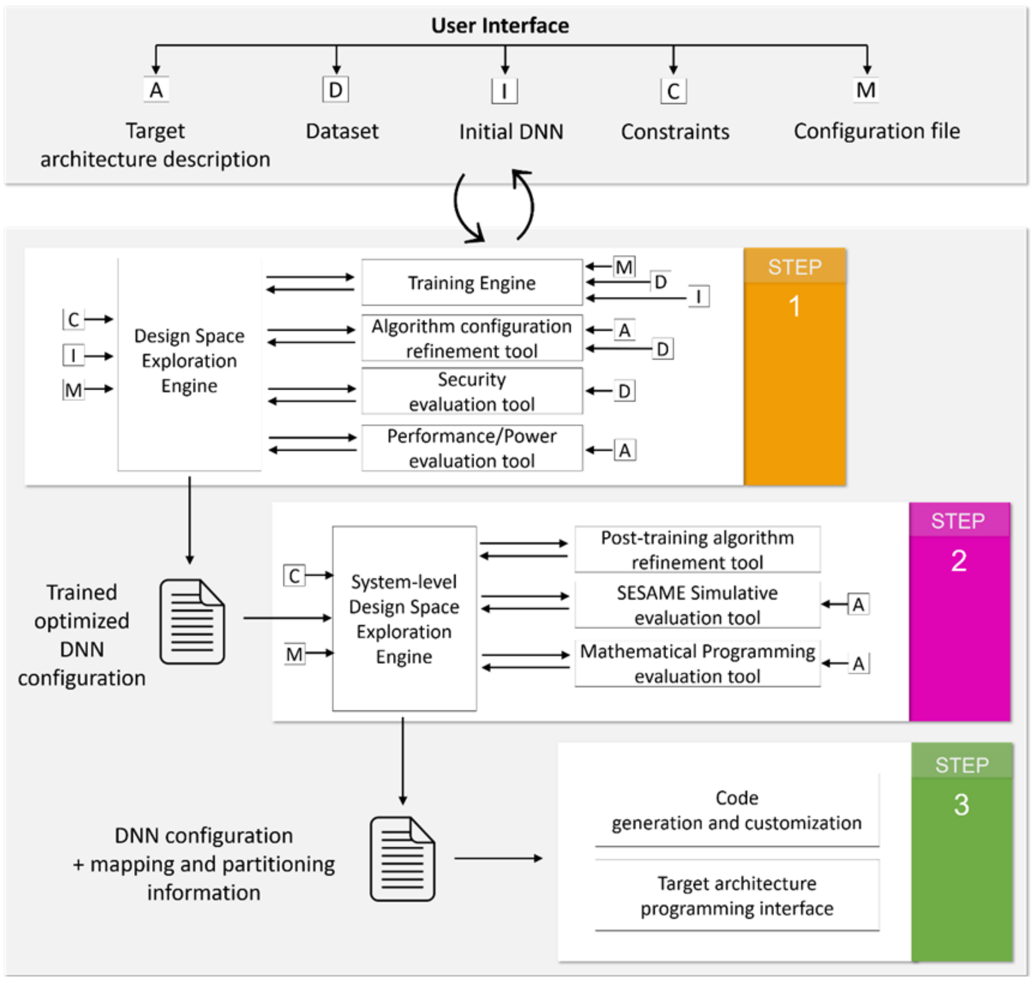

User Interface
A
Target architecture description
D
Dataset
I
Initial DNN
C
Constraints
M
Configuration file
C
I
M
Design Space Exploration Engine
Training Engine
M
D
I
Algorithm configuration refinement tool
A
D
Security evaluation tool
D
Performance/Power evaluation tool
A
STEP 1
Trained optimized DNN configuration
C
M
System-level Design Space Exploration Engine
Post-training algorithm refinement tool
SESAME Simulative evaluation tool
A
Mathematical Programming evaluation tool
A
STEP 2
DNN configuration + mapping and partitioning information
Code generation and customization
Target architecture programming interface
STEP 3

Figure 1: General architecture of the ALOHA framework.

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

The graphic interface guides the user during the definition of a use-case, shows work in progress and provides visualization of results on the implemented steps in the form of graphs and tables. It is developed as web application and provides ease-of-use and accessibility to the tool flow, ensuring future adoption among deep learning practitioners who are not participating in this project.

Each step of the tool flow requires specific interactions, interconnections, and transmission of information among its own components to perform its function. In the next two sections, we provide a short description of each step that is useful for understanding how the framework works, and we present the developed user interface.

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

3 Tool flow environment

This section provides a short description of each steps of the ALOHA tool flow.

Please note that the three steps of the tool flow are, respectively, directly related to the activities performed in work packages *WP2-Automated algorithm design and configuration*, *WP3-Automated application partitioning and mapping*, and *WP4-Hardware abstraction layer and runtime management*, and to their outcomes. A detailed explanation of how each step is performed and how each component is implemented can be found in deliverables D2.2 (Report on automated algorithm configuration - Update), D3.2 (Report on automated application partitioning and mapping - Update), and D4.2 (Report on hardware abstraction layer techniques - Update).

3.1 Step 1 - Automation of the algorithm design process

The first step is guided by a *Design Space Exploration (DSE) engine*, which performs a multi-objective exploration using a genetic algorithm and accesses a set of refinement and evaluation tools to generate the optimal DNN configuration, depending on the target application, the constraints imposed, and the target hardware platform selected by the user (see Figure 1). Each evaluation tool is able to estimate a specific parameter for each design point. The DSE engine starts its operation using an initial DNN configuration provided as input by the user, or generating itself a first population of design points corresponding to random algorithm configurations in terms of number of layers, kernel size, layer connectivity. Then, to refine and evaluate the generated design points the DSE engine cooperates with the following set of satellite tools:

- **Refinement tool for parsimonius inference**. It tries to reduce the computational workload associated with the inference execution of a candidate design point, the size of the memory footprint, and the IO bandwidth requirements, by applying quantization and pruning techniques;
- **Accuracy evaluation tool**. It assesses the accuracy of a candidate design point. This tool is based on a Training Engine, which can be configured to train a DNN model from scratch or to apply transfer learning techniques to it. In the latter mode, the engine tries to reuse pre-trained network models, personalizing them for the target use case. The accuracy evaluation tool offers local hyper-parameter exploration, flexible data parsing (multiple input formats) and flexible use-case configuration (multiple AI tasks, such as classification, detection, tracking).
- **Security evaluation tool**. It evaluates the robustness of a candidate design point to adversarial attacks happening in prediction phase.
- **Performance and Power evaluation tool**. It evaluates the performances and the power consumption associated with the execution of the inference of a candidate design point on a target architecture. It converts the DNN model coming from the DSE engine to an analyzable application model (i.e. a Cyclo-Static DataFlow model). Taking into consideration the target architecture description received as input, this satellite tool allows to pre-estimate what will be the execution time and power consumption of the actors inside the generated model.

When the exploration is finished, and the most suitable design choice is identified, the DSE engine triggers the next step of the tool flow.

3.2 Step 2 - Optimization of the partitioning and mapping

The second step of the tool flow is guided by a *System-level DSE engine*, which identifies the best partitioning of the algorithm configuration generated by step 1 in sub-tasks, and finds the optimal mapping scheme of these sub-tasks on the different processing units available in the target hardware platform, able to satisfy requirements and constraints specified by the user (i.e. throughput, latency and power). Similarly to the

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

previous step, this is done using a genetic algorithm for surfing the design space and requiring evaluation of the candidate partitioning and mapping scheme to two satellite tools: **Sesame** and **Architecture Optimization Workbench** (AOW). These tools simulate computation and task-to-task communication and provide approximates on execution (cycle) times, energy consumption, hardware utilization and resource contention. AOW explores the whole design space using an analytic approach, while Sesame performs more precise simulation over a more limited search space. The System-level DSE engine can also access the **post-training algorithm refinement tool** for parsimonious inference to achieve a workload reduction by considering specific features of the target architecture. This satellite tool tries to reduce the computational workload by applying both a sophisticated on-line data-dependent kernel/component pruning mechanism and a conversion from static to dynamic computing graph to the DNN model.

When the system-level exploration is finished and the more efficient mapping of the DNN configuration is identified, the last step of the tool flow can be carried out.

3.3 Step 3 - Automation of the porting process

The last step of the tool flow is the porting of the DNN configuration on the target hardware architecture. A programming interface receives as input the partitioning and mapping information generated by step 2, and translates them into specific calls to computing and communication primitives exposed by the target hardware architecture. The generated platform-specif code is then customized to reduce as much as possible power consumption and improve performance using, when possible, optimization techniques such as power gating, clock gating and frequency scaling.

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

4 User interface design

The ALOHA user interface is based on a Kaban board, a well-known artifact in the Agile domain, that lets the user understand the status of the different executions in a single view. It has been designed to be as compact and informative as possible while letting users carry out any task they want with the minimum of fuss. The interface provides users with the ability to monitor the state of each step of multiple projects through the tool flow at any time. It is built using a state-of-the-art JavaScript-based framework called ReactJS[3].

The home screen is shown in Figure 2.

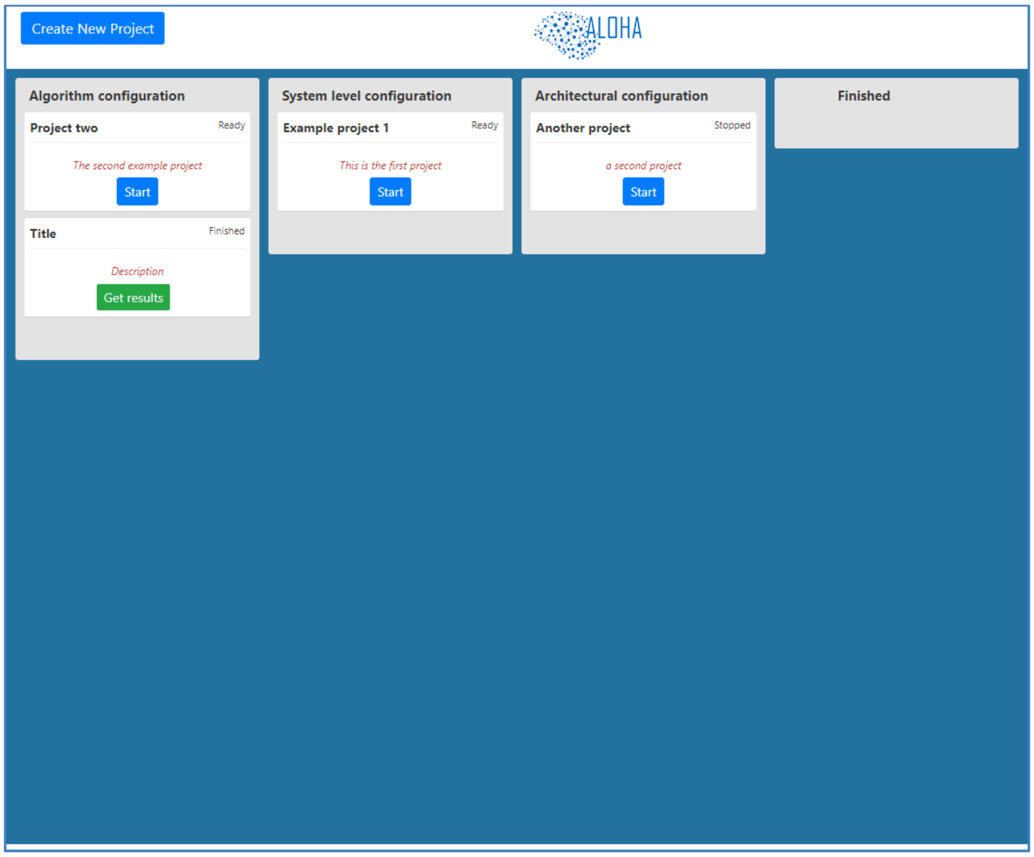


Figure 2: The home screen of the ALOHA user interface

The Kanban approach allows to have a simple and intuitive graphic interface, with only the important information highlighted. The visual board is divided into columns, each of them representing a specific development step of the tool flow process. Each column is populated with cards that represent work items of different types. Cards can be added in a column, deleted, or dragged from one column to the other.

When creating a new project, it automatically creates a card that is placed in the first column. Once it's ready,

[3] https://en.reactjs.org/

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

the user can click the *Start* button to initialize the execution of the step. Once the execution finishes, the user can access a summary of the results and choose wheter to run the step again or run the next stage after dragging the card to the next column. In general, each card will move from left to right until the full workflow is completed and the work is done.

In particular, to execute a new automatic model configuration and porting, the user starts creating and configuring a new project from scratch. This is done by selecting *Create New Project* from the home screen. This opens the dialog shown in Figure 3, which allows to edit a project title and a description, and to specify information about:

- the expected parameters and performances to be met (*Constraints*),
- the target hardware platform to be used (*Architecture*),
- the target application to be implemented (*Algorithm Configuration*).

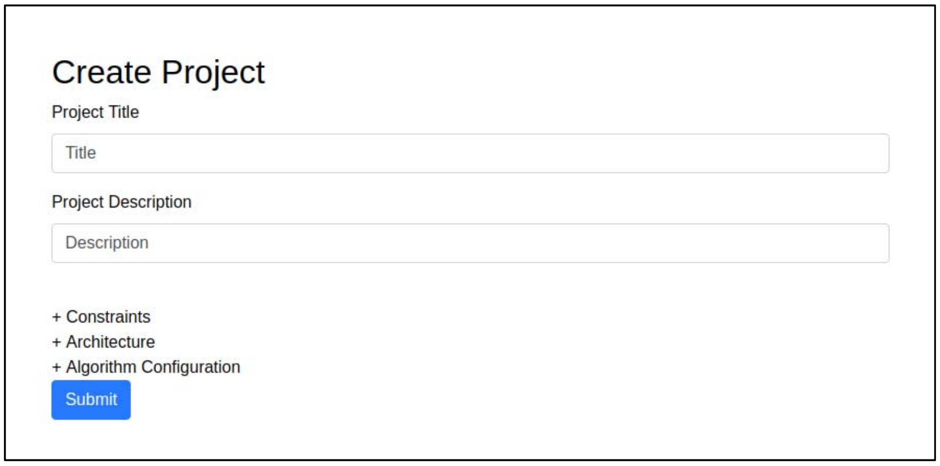
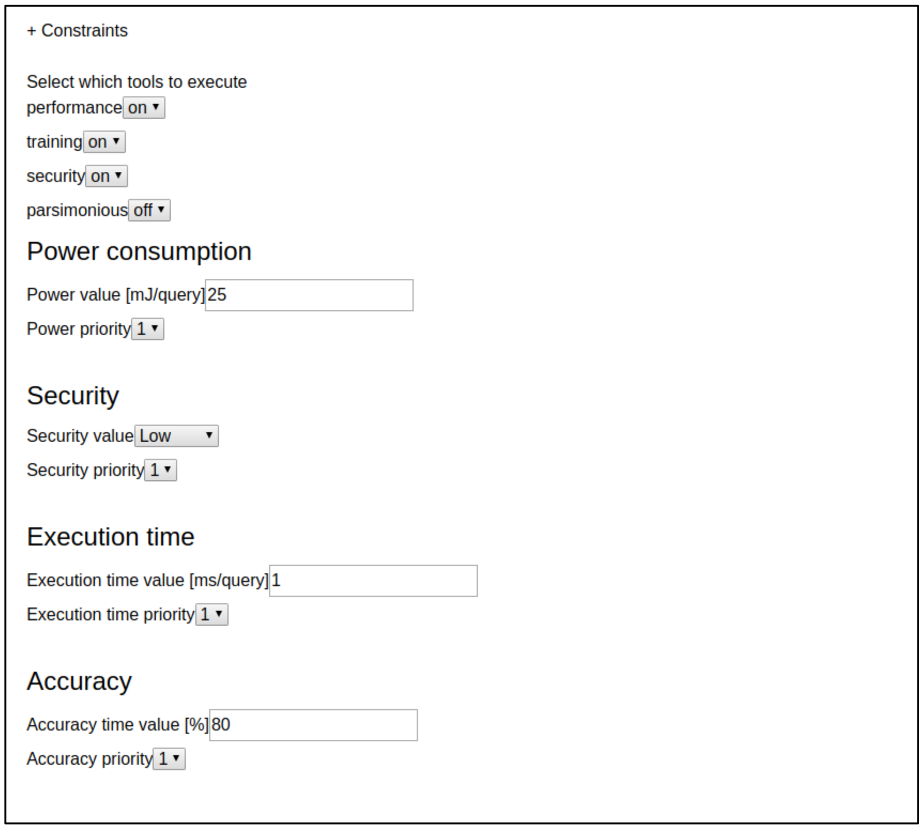


Figure 3: Create a New Project dialog box

The user can define the desired values for accuracy, security, execution time and power consumption in the *Constraints* dialog box shown in Figure 4.

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.



+ Constraints

Select which tools to execute

performance on ▾

training on ▾

security on ▾

parsimonious off ▾

Power consumption

Power value [mJ/query] 25

Power priority 1 ▾

Security

Security value Low ▾

Security priority 1 ▾

Execution time

Execution time value [ms/query] 1

Execution time priority 1 ▾

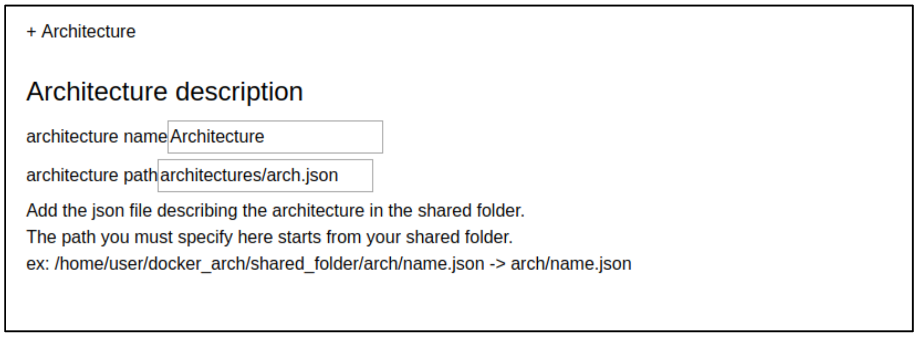
Accuracy

Accuracy time value [%] 80

Accuracy priority 1 ▾

Figure 4: Constraints dialog box

The user can upload an architecture specification file to the project by means the *Architecture* dialog box shown in Figure 5. This .json file provides information about the target hardware platforms in terms of population of computing elements, connectivity, and available operating modes (i.e. data types, working frequency and gating conditions).



+ Architecture

Architecture description

architecture name Architecture

architecture path architectures/arch.json

Add the json file describing the architecture in the shared folder.
The path you must specify here starts from your shared folder.
ex: /home/user/docker_arch/shared_folder/arch/name.json -> arch/name.json

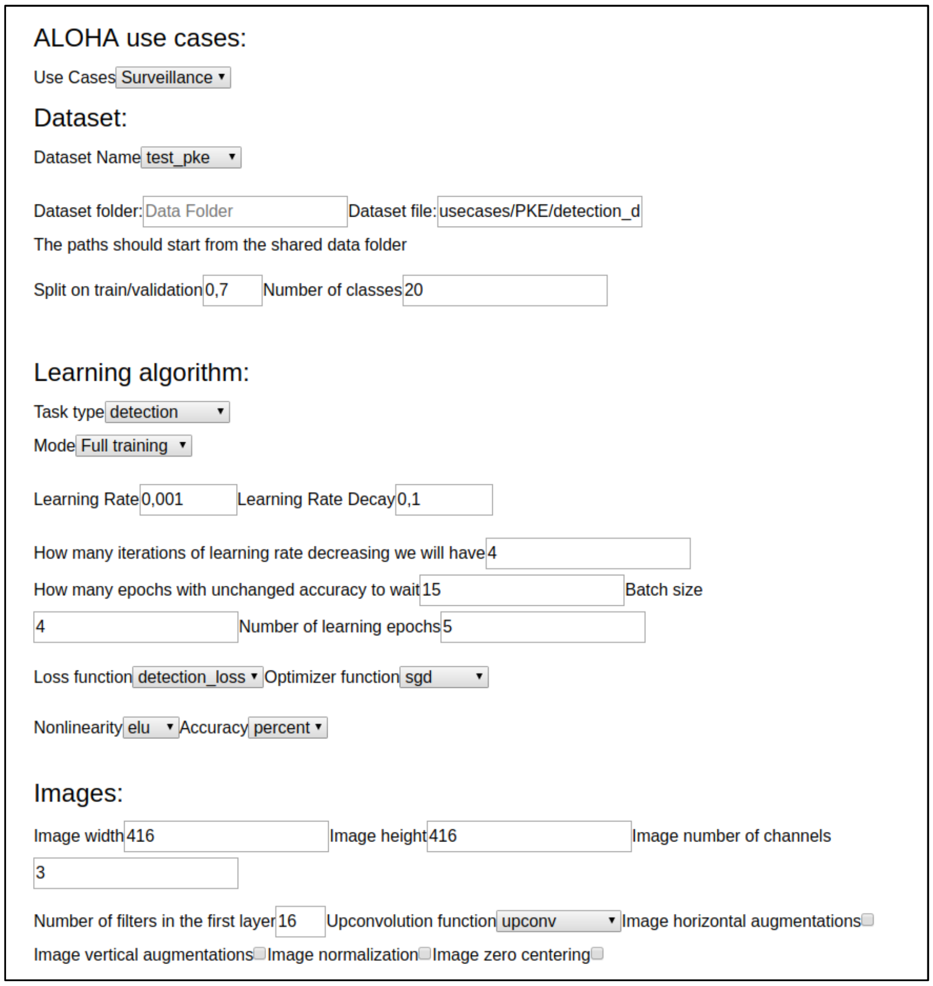
Figure 5: Architecture dialog box

Finally, the user can specify in the *Algorithm Configuration* dialog box (see Figure 6) the target application to be implemented by:

- specifying a link to the shared folder location that stores the dataset to be used; In future implementations this will be updated to a file uploading form.
- uploading an initial DNN configuration;

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

- specifying additional paramaters about the task type, training rate, number of training epochs, batch size, optimization method, loss function



ALOHA use cases:

Use Cases Surveillance

Dataset:

Dataset Name test_pke

Dataset folder: Data Folder Dataset file: usecases/PKE/detection_d

The paths should start from the shared data folder

Split on train/validation 0,7 Number of classes 20

Learning algorithm:

Task type detection

Mode Full training

Learning Rate 0,001 Learning Rate Decay 0,1

How many iterations of learning rate decreasing we will have 4

How many epochs with unchanged accuracy to wait 15 Batch size 4 Number of learning epochs 5

Loss function detection_loss Optimizer function sgd

Nonlinearity elu Accuracy percent

Images:

Image width 416 Image height 416 Image number of channels 3

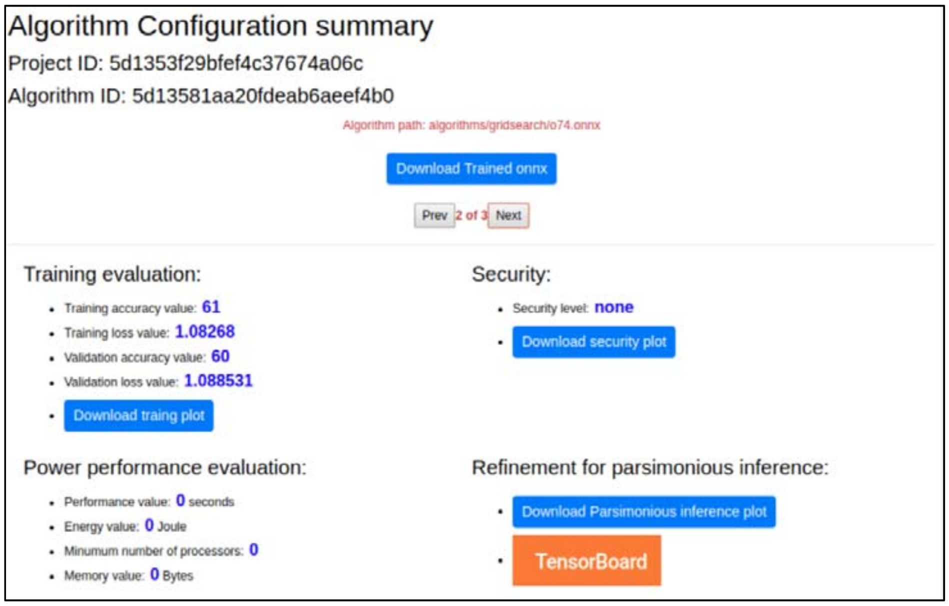
Number of filters in the first layer 16 Upconvolution function upconv Image horizontal augmentations ☐

Image vertical augmentations ☐ Image normalization ☐ Image zero centering ☐

Figure 6: Example of Surveillance use-case specification

Once that all the information has been provided, the user can run the project by selecting *Submit* in the Create project dialog box (see Figure 3) what automatically creates a new card for it. By clicking the *Start button* the execution of the first step in the flow is launched. When the process corresponding to this first step is terminated, the option *Get result* is available in the card and the Results dialog box is shown in the screen. TensorBoard-compliant logs are also gathered to enable joint visualization of pareto points and exploration results. When each of the steps is completed, the project card can be dragged to the next phase (see Figure 2) and results can be obtained in a similar way.

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.


Algorithm Configuration summary

Project ID: 5d1353f29bfef4c37674a06c

Algorithm ID: 5d13581aa20fdeab6aeef4b0

Algorithm path: algorithms/gridsearch/o74.onnx

Training evaluation:

- Training accuracy value: 61
- Training loss value: 1.08268
- Validation accuracy value: 60
- Validation loss value: 1.088531

Security:

- Security level: none

Power performance evaluation:

- Performance value: 0 seconds
- Energy value: 0 Joule
- Minumum number of processors: 0
- Memory value: 0 Bytes

Refinement for parsimonious inference:

Figure 7: Example of Results dialog box reporting the results for the algorithm configuration step

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

5 Integration process

As already described in deliverable D1.1 (Report on general specifications and interface definition), the ALOHA project follows an Agile software development approach based on the concept of microservices. Each step of the ALOHA tool flow has been broken into smaller, completely independent microservices components (see Figure 1), which can be built, updated and deployed individually to implement a single, defined purpose or task. This microservice approach ensures flexibility, allows the project developers to work autonomously and make changes independently, and allows the usage of different programming languages for each component. A failure/fault in one microservice does not affect the functioning of other components of the tool flow. Each component can be extended or replaced with less effort compared to a monolithic architecture.

This section provides a detailed insight into the integration process of the project.

5.1 Building the microservice architecture

The main component of the designed architecture are microservices. Each microservice is used to create loosely isolated running environments, so that each component can be independently built and deployed to implement a specific feature of the ALOHA tool flow. In particular, Docker [4] has been used to implement, manage and execute the images of each of the microservices as Docker images or containers.

Each individual component of the ALOHA the tool flow has an individual project in the ALOHA's Gitlab repository https://gitlab.com/aloha.eu.

To integrate all the projects into a single one, corresponding to the complete toolflow, a new project was created, available at https://gitlab.com/aloha.eu/docker_arch, containing an integrated version that includes scripts and dependency lists needed to:

a) clone all the individual component repositories;

b) compose a docker environment including all the individual component containers;

c) bring up a front-end instance with available GUI;

d) bring up the database facility needed for the project.

A graphic description of the implemented architecture is shown in Figure 8. It shows the interaction among the different components of the ALOHA toolflow. The whole architecture is designed to run in a distributed environment. The user can access the toolflow through the web GUI provided by the front-end orchestrator. By using it, the user is able to interact with the toolflow and action launched in the frontend is then communicated to the back-end orchestrator which executes the corresponding tools.

The components run on docker containers, so they are independent and they communicate through REST APIs, as it was decided when creating the design of the architecture. To enable parallel computation, each multiple workers are allowed within each component. The synchronization among the component and its

[4] https://www.docker.com/

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

workers is implemented by the RQ Python library based on the lightweight Redis DB[5].

The toolflow produces different types of files, such as ONNX [6] files containing the description of each generated models, TensorBoard logs, text logs, which are stored in a shared data folder accessible by each component. A shared MongoDB [7] database is used to store, manage and facilitate the access from all the ALOHA modules to other types of information, i.e. constraints, training hyper parameters, security level, performance evaluation, stored in separated MongoDB collections.

The GUI provides the access to TensorBoard for each existing project which plots training graphs and pareto representations of the produced evaluations.

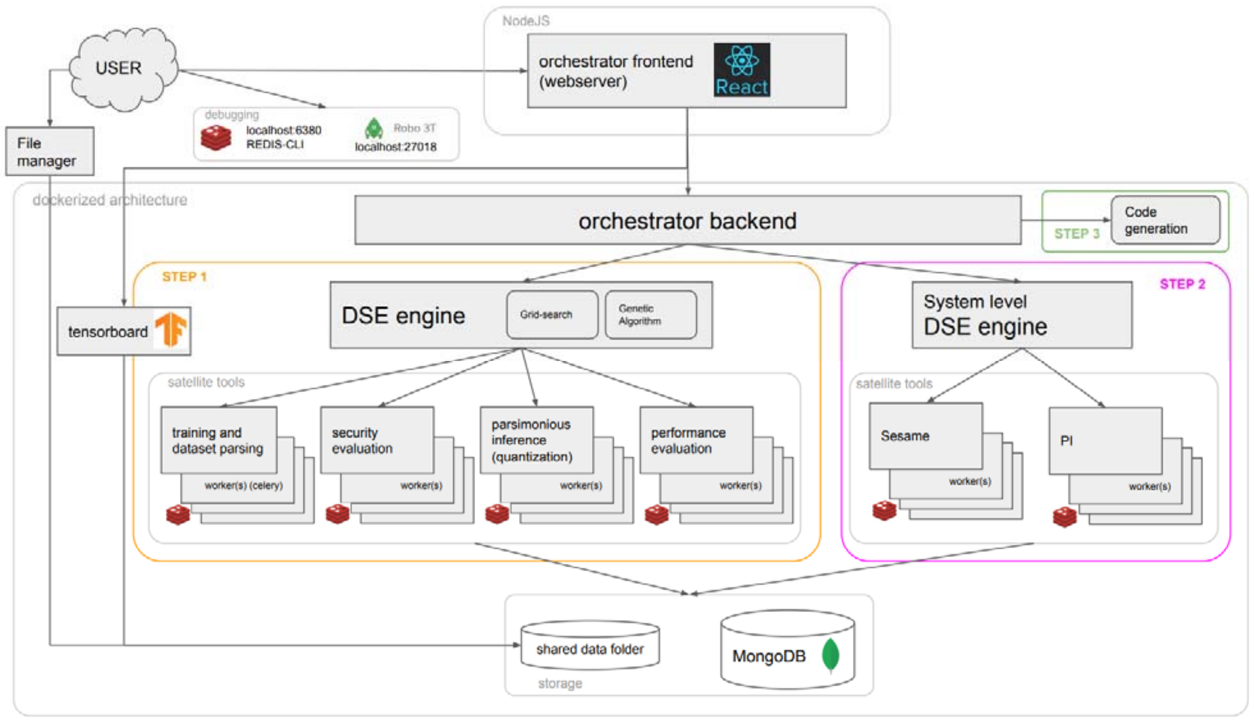


Figure 8: ALOHA microservices architecture

5 https://redis.io/

6 https://onnx.ai/

7 https://www.mongodb.com/

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

6 Deployment and integration testing process

All the components have been tested in their standalone version, as well as inside the integrated environment using a simple generic benchmark based on the CIFAR dataset and a basic image classification task. The implementation of the three ALOHA use-cases is now supported and ready to be used as stress-tests for the toolflow. For more information please refer to Deliverables D5.2 (Report on Use Case Implementation First Update), D5.4 (First release of use-case_implementation) and D5.6 (Use-case demonstrators).

Moreover, in order to verify easiness of ALOHA toolflow usage, and therefore the attractiveness of the solution for prospective customers and stakeholders, which is one of the prerequisites of our exploitation plan, the Consortium has included in the deployment and integration testing process, a co-design session with the use-case providers. The purpose of this session was to gather feedback and opinions on the current version of the Guidelines for interacting with the ALOHA toolflow provided by technical partners on the ALOHA GitLab repository. The technical partners of the Consortium have prepared such guidelines to speed up the use-case implementation processes.

The session has been held in Vienna at month M17. After reading the Guidelines, use-case providers had the possibility to perform a first test phase. The use-case datasets and tasks have been provided in input to the toolflow, correctly parsed and passed through the execution of the specifically required components. Use-case specific data parsers and CNN actors (loss functions, convolution/deconvolution actors, ONNX representations, etc…) have been implemented and made available on the repository to be used in all the components.

During the testing session, technical partners from UNICA and SCCH were sitting close to use-providers, gathering feedback and taking notes on what they were experiencing, what they liked and disliked about it, what their expectations were, and what they were trying to do with the available demo.

After the testing session, ALOHA use-providers provided useful and constructive critiques of the GUI and suggestions on possible enhacements. Some of the technical issues reported during the session were addressed during the two-days meeting, while suggestions and new ideas have been collected and will be implemented in the GUI by technical partners in the next months.

At month M18, two main issues – based on use-providers feedback – have been addressed and are included in the GUI:

- Possibility to delete a project directly from the GUI – this functionality has been added and is now available, as shown in Figure 9.
- Possibility to have default hyper-parameters for each use-case - this functionality has been added and is now available.

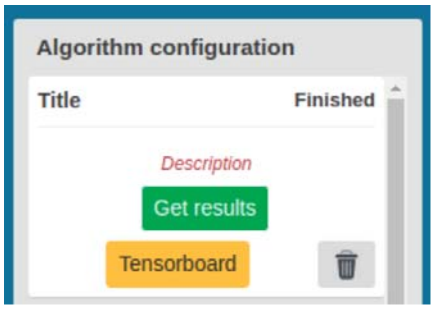


Figure 9: Screenshot of the GUI showing the delete project option

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

7 ALOHA Technical KPIs

During the specification phase of the project, ALOHA partners have identified a list of technical Key Performance Indicators (KPIs) to be used for the assessment of the toolflow, see Deliverable D1.1.

An updated version of the KPIs list is presented below. The following updates have been introduced in comparison to D1.1:

- **KPIs for security level** (KPI1_4 and KPI1_5)
- **KPIs for performance level** (KPI1_6 and KPI1_7)

KPIs will be evaluated comparing with design flows previously available for the use cases before the adoption of ALOHA. The list will evolve and gain more precision and substance during the lifespan of the project.

Table 1: ALOHA KPIs list

KPI1_1 - Time spent in algorithm design	**Surveillance UC**	**Smart Industry UC**	**Medical UC**
Without adoption of ALOHA: Effort required in manual flow months/weeks	Fixed designed by hand topology. Small, few layers, trained with in house procedure. Estimated time for a new design: months	Not available - limited experience in DL development – Estimated time for a new speech-processing software project: months	Composition of multiple ML and image processing modules. Estimated time for a new design: months
With ALOHA: Effort required days	More complex, custom tuned topology. Estimated time for a new design: days	Custom tuned topology. Estimated time for a new design: days	More compact composition. Estimated time for a new design: days
Comments: this takes into account both manual efforts as experienced in previously available design flows on network configurations and training procedures comparable with most common literature cases. Network configurations and training procedures can have significant impact on the execution time of ALOHA building blocks requiring training tasks, on commonly available training facilities.			
KPI1_2 - Time spent in parallel application porting and optimal mapping onto MPSoCs	**Surveillance UC**	**Smart Industry UC**	**Medical UC**
Without adoption of ALOHA: Effort required in manual flow months/weeks	Not available: application running on servers/workstations	Not available for DL. Estimated time needed for optimal mapping of an embedded application on target plaform: weeks	Not available: application running on servers/workstations
With ALOHA: Effort required hours	Application optimally mapped on target platform in hours	Application optimally mapped on target platform in hours	Application optimally mapped on target platform in hours
Comments: this takes into account both manual efforts as experienced in previously available design flows on network configurations comparable with most common literature cases (10-50 layers) and common MPSoC sizes (around 5-10 processors). Network configurations and design space size (number of possible mappings) can have significant impact on the execution time of the system-level desing process.			

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

KPI1_3 - Time spent in implementation of DL code on customized computing platforms	**Surveillance UC**	**Smart Industry UC**	**Medical UC**
Without adoption of ALOHA: Effort required in manual flow: days/month (accounting for the use of existing commercial support tools)	Not available: application running on servers/workstations	Not available for DL. Expected time needed for implementation on heterogeneous platforms: weeks	Not available: application running on servers/workstations
With ALOHA: Effort required hour/days (accounting for the use of existing commercial support tools)	Automated middleware code generation in hours (requires a preliminary adaptation of generic primitives to platform-specific primitives)	Automated middleware code generation in hours (requires a preliminary adaptation of generic primitives to platform-specific primitives)	Automated middleware code generation in hours (requires a preliminary adaptation of generic primitives to platform-specific primitives)
Comments: effort required in current toolflow depends on the complexity of the target platform programming model. With ALOHA, the first toolflow execution requires adaptation of the generated middleware to the target programming model. Other executions will require less effort, exploiting already available adaptation.			
KPI2_1 - Energy efficiency gain	**Surveillance UC**	**Smart Industry UC**	**Medical UC**
Without adoption of ALOHA: full power consumption and memory footprint of a "regular" DNN	Hand-crafted CNN topology, workload reduction achieved by network size reduction.	Not available. No current procedure for DL development. Options limited to publicly available code.	Hand-crafted CNN topology. High complexity for high accuracy.
With ALOHA: automated pruning and quantization. We can expect reduction of workload of around 30%-50% and the consequent power-consumption reduction	Custom automatically generated CNN topology, mappable on embedded hardware.	Custom automatically generated CNN topology, mappable on embedded hardware.	Custom automatically generated CNN topology, mappable on embedded hardware.
Comments: the expected level of reduction reported here derives from general assumptions made on experiments performed starting from well-known "regular" algorithms, after pruning and quantization. Han et al.[8] report an average energy gain of 4x via compression/pruning of DNN layers on an embedded GPU. Moons et al.[9] report ~10x improvement in energy with little accuracy loss when using HW optimized for quantizaiton, which however accounts only for computational power. 30-50% keeps into account these and similar results, the fact that parts of the DNN might not be quantized/pruned, the cost of overheads which do not scale with precision (such as control) or scale less than computation (e.g. data movement scales down ~linearly with precision, whereas data computation scales down ~quadratically with precision).			

[8] S. Han et al., "EIE: Efficient Inference Engine on Compressed Deep Neural Network," in Proceedings of the 43rd International Symposium on Computer Architecture, Piscataway, NJ, USA, 2016, pp. 243–254.

[9] B. Moons, K. Goetschalckx, N. Van Berckelaer, and M. Verhelst, "Minimum Energy Quantized Neural Networks," arXiv:1711.00215 [cs], Nov. 2017.

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

KPI1_4- Time spent for security evaluation	**Surveillance UC**	**Smart Industry UC**	**Medical UC**
Without adoption of ALOHA: weeks/months (accounting for the use of existing open-source/ commercial support tools)	No framework available. Required integration with existing open-source/commercial support tools to create adversarial/perturbed video frames. Test with increasing levels of perturbation, for each frame of each video. Estimated time for a full evaluation: weeks/months.	No framework available. Required integration with existing open-source/commercial support tools to create adversarial/perturbed audio samples. Need of a pervasive evaluation in the transformed space of the spectrogram. Estimated time for a full evaluation: weeks.	Required integration with existing open-source/commercial support tools to create adversarial/perturbed images. Evaluation with increasing levels of perturbation, on the whole dataset. Estimated time for a full evaluation: weeks.
With ALOHA: hours/days	Automatic creation of adversarial/perturbed examples. Reproducible tests and reporting. Possibility of performing slow/fast evaluations. Estimated time for a full evaluation: minutes/hours.	Automatic creation of adversarial/perturbed examples. Reproducible tests and reporting. Possibility of performing slow/fast evaluations. Estimated time for a full evaluation: minutes/hours.	Automatic creation of adversarial/perturbed examples. Reproducible tests and reporting. Possibility of performing slow/fast evaluations. Estimated time for a full evaluation: minutes/hours.
Comments: two different evaluation modes have been defined for performing security evaluation of deep learning algorithms in ALOHA. The “slow evaluation” peforms an in-depth analysis of the model, inspecting a large number of adversarial scenarios (maximum perturbation allowed, number of attack points), to provide an accurate security/robustness evaluation of the model at hand. The “fast evaluation” aims to estimate the security/robustness of a model by using fewer evaluation scenarios, to provide a faster estimate. This mode can speed up the evaluation process up to 10 times, while retaining sufficiently-precise security-level estimates.			

KPI1_5 – Time spent for improving security/robustness	**Surveillance UC**	**Smart Industry UC**	**Medical UC**
Without adoption of ALOHA: weeks/months (accounting for the use of existing open-source/ commercial support tools)	No framework available. Required integration with existing open-source/commercial support tools to implement mitigation/defensive strategies. Estimated time for development/testing: weeks/months.	No framework available. Required integration with existing open-source/commercial support tools to implement mitigation/defensive strategies. Estimated time for development/testing: weeks/months.	No framework available. Required integration with existing open-source/commercial support tools to implement mitigation/defensive strategies. Estimated time for development/testing: weeks/months.
With ALOHA: hours/days	Mitigation/defensive strategies are implemented within the toolflow. Reproducible tests and reporting.	Mitigation/defensive strategies are implemented within the toolflow. Reproducible tests and reporting.	Mitigation/defensive strategies are implemented within the toolflow. Reproducible tests and reporting.

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

	Estimated time for development/testing: hours/days.	Estimated time for development/testing: hours/days.	Estimated time for development/testing: hours/days.

Comments: two different defensive/mitigation strategies are envisioned in ALOHA. The former is based on *adversarial training*, namely, on the idea of retraining the model on the adversarial/perturbed samples. The second is based on the development of ad-hoc regularized loss functions to be optimized during training, to improve model robustness against specific kinds of input data perturbation.

KPI1_6– Performance/power evaluation accuracy	**Surveillance UC**	**Smart Industry UC**	**Medical UC**
Without adoption of ALOHA: identifies order of magnitude	Considers only number of operations or memory footprint as a method for estimating performance and power consumption.	Considers only number of operations or memory footprint as a method for estimating performance and power consumption.	Considers only number of operations or memory footprint as a method for estimating performance and power consumption.
With ALOHA: less than 20% inaccuracy	Relies on detailed architecture model to estimate performance and power consumpion.	Relies on detailed architecture model to estimate performance and power consumpion.	Relies on detailed architecture model to estimate performance and power consumpion.

Comments: ALOHA integrates three tools (perfromance evaluation in WP2, Sesame and AOW) for accurate estimation of performance, memory accesses and concurrent processing. Parallel execution on different cores, communication, synchronization and memory subsystems are modeled and considered when comparing different designs and mapping.

KPI1_7 – Time spent for accurate performance/power evaluation	**Surveillance UC**	**Smart Industry UC**	**Medical UC**
Without adoption of ALOHA: weeks/months	Often a precise estimate is obtained only after preliminary runs on the actual target platform, requiring significant prototyping effort.	Often a precise estimate is obtained only after preliminary runs on the actual target platform, requiring significant prototyping effort.	Often a precise estimate is obtained only after preliminary runs on the actual target platform, requiring significant prototyping effort.
With ALOHA: hours	Relies on detailed architecture model to estimate performance and power consumpion.	Relies on detailed architecture model to estimate performance and power consumpion.	Relies on detailed architecture model to estimate performance and power consumpion.

Comments: ALOHA pre-characterized models (at different levels) for accurate estimation of performance and power. This enables to take power- and performance-aware relevant design choices and customization decisions early in the design flow, without actual trials of prototype code on real hardware.

Copyright - This document has been produced and funded under the ALOHA H2020 Grant Agreement 780788.
Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the ALOHA Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.