



Human-centric algebraic machine learning

Overall Architecture Definition - Final M54

Deliverable D1.4

**WP1 - Project Management,
Architecture and Technical Coordination**



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 952091.

Consortium



ALGEBRAIC AI



Chantalimaud
Foundation



EPROSIMA
The Middleware Experts



German
Research Center
for Artificial
Intelligence



TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

uc3m



Project

Title: ALMA: Human Centric Algebraic Machine Learning
 Acronym: ALMA
 Coordinator: Proyectos y Sistemas de Mantenimiento SL (eProsima)
 Grant agreement: 952091
 Call: H2020-EIC-FETPROACT-2019
 Program: Horizon 2020
 Start: 01 September 2020
 Duration: 54 months
 Website: <https://alma-ai.eu>
 E-mail: alma@eprosima.com
 Consortium: **Proyectos y Sistemas de Mantenimiento SL (eProsima) (EPROS)**, Spain
Champalimaud Foundation (CF), Portugal
ALGEBRAIC AI (ALG), Spain
German Research Center for Artificial Intelligence (DFKI), Germany
Inria (Inria), France
TU Kaiserslautern (TUK), Germany
FIWARE Foundation e.V. (FF), Germany
Technical Research Centre of Finland (VTT), Finland
Carlos III University (UC3M), Spain

Deliverable

Number: **D1.4**
 Title: **Overall Architecture Definition - Final M54**
 Month: 54
 Work Package: WP1 - Project Management, Architecture and Technical Coordination
 Work Package leader: Proyectos y Sistemas de Mantenimiento SL (eProsima)
 Deliverable leader: German Research Center for Artificial Intelligence (DFKI)
 Deliverable type: Report (R)
 Dissemination level: Public (PU)
 Date of submission: 28/02/2025
 Version: v4.6
 Status: Finished

Version history

Version	Date	Responsible	Author/Reviewer	Comments
v1.0	19/05/2021	EPROS	Javier Paris	D1.1 - First draft
v1.1	20/05/2021	EPROS	Raúl Sánchez-Mateos Lizano	D1.1 - Second draft
v1.2	24/05/2021	ALG	Nabil Abderrahaman	D1.1 - Third draft
v1.3	25/05/2021	EPROS	Raúl Sánchez-Mateos Lizano	D1.1 - First Review (RV1)
v1.4	27/05/2021	CF	Gonzalo G. de Polavieja	D1.1 - Second Review (RV2)
v2.0	15/02/2022	EPROS	Juan López Fernández	D1.2 - First draft
v2.1	28/02/2022	EPROS	Raúl Sánchez-Mateos Lizano	D1.2 - Revision (RV1)
v2.2	29/06/2022	EPROS	Javier Paris Uhryn	Apply PO and EU Reviewers suggestions
v2.3	01/07/2022	EPROS	Juan López Fernández	D1.2 - Revision (RV2)
v3.0	10/09/2023	EPROS	Irene Bandera Moreno	D1.3 - First draft
v3.1	15/09/2023	EPROS	Raúl Sánchez-Mateos	D1.3 - First review & ALMA overview image
v3.2	20/09/2023	EPROS	Carlos Ferreira González	D1.3 - ALMA overview
v3.3	20/09/2023	EPROS	Raúl Sánchez-Mateos	D1.3 - Second review and apply internal suggestions
v4.0	08/02/2025	EPROS	Denisa Alexandru	D1.4 - First draft
v4.1	17/02/2025	DFKI	Vitor Fortes	D1.4 - Scientific contribution draft
v4.2	17/02/2025	ALG	Nabil Abderrahaman	D1.4 - Scientific contribution
v4.3	27/02/2025	EPROS	Raúl Sánchez-Mateos	D1.4 - Scientific contribution, final review and proofreading
v4.4	27/02/2025	ALG/CF	Fernando Martín-Maroto	D1.4 - Scientific contribution
v4.5	27/02/2025	VTT	Stephen Fox	D1.4 - Scientific contribution
v4.6	27/02/2025	EPROS	Raúl Sánchez-Mateos	D1.4 - Executive summary and introduction

Executive summary

Algebraic Machine Learning (AML) is a novel machine-learning technique based on algebraic representations of data. Unlike statistical learning, AML algorithms are robust regarding the statistical properties of the data and are parameter-free. AML systems can retain what they learn, including sociocultural ethical practices for human–artificial intelligence systems. The EU-funded ALMA project aims to study the mathematical properties of AML and to leverage them to develop a new generation of interactive human-centric machine-learning systems. These systems are intended to minimize bias to facilitate trust while better enabling distributed collaborative learning.

This report corresponds to *Deliverable D1.4 - Overall Architecture Definition - Final M54* of the ALMA project. This deliverable provides an in-depth exposition of the architecture supporting the AML ecosystem. It details the design and integration of key components, including the AML Engine, which processes data to build AML models; the AML Description Language (AML-DL), which offers a concise, algebraically driven method for problem specification; and the AML Consistency Checker (AML-CC), ensuring that the encoded relations are logically coherent. Central to this architecture is the AML Integrating Platform (AML-IP), a sophisticated communication framework that leverages decentralized, peer-to-peer interactions via protocols like the Data Distribution Service (DDS). This platform facilitates distributed, asynchronous learning across multiple nodes, enabling scalable and collaborative model training.

Additionally, the document introduces the main scientific contributions of the ALMA project which encapsulates the extensive research outputs generated by the consortium. This section provides an overview of publications and theoretical advancements—ranging from the development of atomized semilattices to innovative methods for semantic embedding and real-world applications such as image classification and complex combinatorial problem-solving. These contributions not only prove the technical quality of AML but also highlight its potential to integrate ethical and sociocultural dimensions into machine-learning systems.

By combining a transparent, distributed architecture with cutting-edge theoretical and practical contributions, ALMA lays the foundation for next-generation AI systems that are not only more efficient and scalable but also more interpretable and ethically aligned. The design is further enhanced by its compatibility with established platforms such as ROS 2 and FIWARE, positioning AML as a versatile solution for a broad range of applications in both research and industry.

Revision changes M18

Revision of deliverable *D1.1 - Overall Architecture Definition - Initial* for *Work Package 1: Project Management, Architecture and Technical Coordination*.

- Correction of typos.
- In subsection 3.1, *parallel learning* is renamed to *collaborative learning* in order to remain consistent with *Deliverable D6.4 - AML-IP implementation - Final M48*.
- Section 7 has been extended with a brief explanation of the design made for the AML-IP in a *workload distribution scenario* (subsection 7.1).
- Updated AML Integrating Platform (AML-IP) communication and nodes layer model present in subsection 6.2, now including the newly introduced AML-IP Algebraic Main Node (AML-IP-Main) and AML-IP Algebraic Computing Node (AML-IP-Computing).
- *eProxima Integration Service* has been replaced by the *DDS Router* as the chosen tool for bridging the connection between different Local Area Networks (LANs).

Revision changes M36

Revision of deliverable *D1.2 - Overall Architecture Definition - M18* for *Work Package 1: Project Management, Architecture and Technical Coordination*.

- AML-IP Mediation Nodes have been introduced to transform data between different formats, offloading this task from AML-IP Algebraic Nodes.
- Section 7 has been updated with a brief subsection of the design made for the AML-IP scenarios, removing the old AML-IP prototype design, no longer required to understand the final architecture.
- Updated AML-IP communication and nodes layer model present in subsection 6.2, now including the newly introduced nodes.

Revision changes M48

Revision of deliverable *D1.3 - Overall Architecture Definition - M36* for *Work Package 1: Project Management, Architecture and Technical Coordination*.

- AML Description Language (AML-DL) description has been updated to include the latest related deliverables.
- New section on scientific contributions. The AML project has resulted in multiple publications, including some under peer review, covering topics such as advancements in atomized semilattices, applications, and model interpretability.
- The Fiware Node has been introduced in AML-IP. This type of node interacts with the FIWARE context broker to manage inference data. It offers mechanisms for retrieving data from a context broker entity, soliciting inference from an Inference Node, and transmitting the inference to the context broker entity.
- Section 7 has been updated with a new scenario about the interaction with the FIWARE Context Broker.
- New Tools have been added, integrating AML-IP with other ALMA technologies. AML Toolkit is a set of software tools that allows users to develop, train, and deploy Algebraic Machine Learning (AML) models. AML Toolkit includes AML-IP and AML-Dashboard. AML Dashboard serves as a user-friendly tool for training, evaluating, and applying AML models to several scenarios. It allows the users to manage AML-IP nodes, perform workload distribution, collaborative learning and distributed inference within the AML-IP framework, as well as monitor the state of the network.

Contents

Executive Summary	4
Revision Changes M18	5
Revision Changes M36	6
Revision Changes M48	7
Contents	8
Acronyms	9
1 Introduction	10
1.1 Purpose of document	10
1.2 Partners involved	11
2 ALMA Project Overview	11
3 Algebraic Machine Learning	13
3.1 AML environment	13
3.1.1 Communication between AML nodes	13
4 Scientific Contributions	15
5 AML Description Language	16
6 AML Integrating Platform	16
6.1 Dependencies	16
6.2 AML-IP Design	17
6.3 AML-IP functional operation	19
6.3.1 Phases of communication	19
6.3.2 AML-IP operational area networks	19
7 Current State	20
7.1 AML-IP workload distribution scenario	21
7.2 AML-IP collaborative learning scenario	22
7.3 AML-IP distributed inference scenario	23
7.4 AML-IP monitor network state scenario	24
7.5 Context Broker Integration	25
8 AML-IP Use Cases	26
8.1 Training MNIST	26
8.2 AML-Dashboard	26
8.3 AML-Toolkit	28
9 Innovation Management	28
10 Conclusions	29
References	31

Acronyms

AI	Artificial Intelligence.
AML	Algebraic Machine Learning.
AML-CC	AML Consistency Checker.
AML-DL	AML Description Language.
AML-Engine	AML Engine.
AML-IP	AML Integrating Platform.
AML-IP-Agent	AML-IP Agent Node.
AML-IP-Alg	AML-IP Algebraic Node.
AML-IP-Computing	AML-IP Algebraic Computing Node.
AML-IP-D	AML-IP Discovery Node.
AML-IP-Edge	AML-IP Algebraic Edge Node.
AML-IP-Engine	AML-IP Algebraic Engine Node.
AML-IP-Inference	AML-IP Algebraic Inference Node.
AML-IP-Main	AML-IP Algebraic Main Node.
AML-IP-Mediation	AML-IP Mediation Node.
AML-IP-MMReceiver	AML-IP Algebraic Model Manager Receiver Node.
AML-IP-MMSender	AML-IP Algebraic Model Manager Sender Node.
AML-IP-Status	AML-IP Algebraic Status Node.
DDS	Data Distribution Service.
GA	Grant Agreement.
LAN	Local Area Network.
ML	Machine Learning.
MSCP	MultiService Communication Protocol.
NAT	Network Address Translation.
P2P	Peer to Peer.
PO	Project Officer.
TCP	Transmission Control Protocol.
WAN	Wide Area Network.
WP	Work Package.
WWW	World Wide Web.

1 Introduction

The ALMA research and innovation project aims to develop a novel approach to the Artificial Intelligence (AI) paradigm by implementing and further exploring Algebraic Machine Learning (AML), a new Machine Learning (ML) technique with a strong foundation in Abstract Algebra and Model Theory. The goal of the EU-funded ALMA project is to leverage the properties of AML to develop a new generation of interactive, human-centred machine learning systems.

1.1 Purpose of document

This document aims to describe the implementation and integration of a platform for the massively distributed parallel execution of AML algorithms. In particular, this document focuses on the Overall Architecture of the project. Here, a summary of the components, how they are connected for sharing data, and the current state of the project are presented. The main elements that will form the architecture for executing AML are the AML Engine (AML-Engine) itself, the AML Description Language (AML-DL), the AML Consistency Checker (AML-CC), and the AML Integrating Platform (AML-IP).

The AML Engine (AML-Engine) carries out the learning process, creating models from data and formal knowledge. It executes the AML algorithm, increasingly building the model as it learns and generalises knowledge from examples and pre-established rules. Several AML-Engine can work on the same problem in parallel, independently building partial models and periodically sharing knowledge. The final model can then be built by aggregating and combining all partial models.

The main goal of AML-DL is to provide the users with a language to describe ML problems for AML, using algebraic semantic embeddings, in a concise and expressive manner. An embedding is a set of positive and negative relations between pairs of elements, that constitute the input data for the AML-Engine. AML-DL and an overview of semantic embeddings can be found in *Deliverable D3.4 - AML-DL Specification - Final M48*.

The AML Consistency Checker (AML-CC) verifies that a set of relations describing an AML problem are all consistent and there are no two rules contradicting each other. This is a necessary step before transferring data to an engine. In essence, a training batch cannot contain two relations of the form $A \text{ equal } B$ and $A \text{ not equal to } B$. However, note that this only affects examples within the same batch, the dataset can contain any number of errors, contradictions and mislabels, as long as they are computed in different training batches. The AML-CC is further explained in *Deliverable D3.9 - AML-DL Consistency Checker - Final M36*.

The AML-IP is the platform that manages the whole communication process between the different nodes that form the AML environment. This platform creates an Internet-wide network and handles communication between nodes in an efficient and distributed way, allowing the creation of a parallel, scalable AML architecture. It also offers a user-friendly framework to easily create AML internal or external networks. This element is explained in section 6, and more details about its design and its actual state can be found in this project's deliverable *Deliverable D6.4 - AML-IP implementation - Final M48*.

This document also includes a description of the main scientific contributions, which serves as an introduction to the scientific and innovative foundations supporting the ALMA project. In this section, readers will find a concise overview of the groundbreaking research outputs—from novel theoretical developments and mathematical models to experimental validations and practical applications—that have been achieved throughout the project. By presenting these contributions, the document not only outlines the technical and architectural aspects of the AML environment but also highlights the significant academic and practical advancements performed within the ALMA project.

1.2 Partners involved

Deliverable Leader:

- German Research Center for Artificial Intelligence (DFKI), Germany

Partners:

- Algebraic AI (ALG), Spain
- Proyectos y Sistemas de Mantenimiento SL (eProsima) (EPROS), Spain

2 ALMA Project Overview

Algebraic machine learning (AML) is a relatively new machine learning technique based on algebraic representations of data. Unlike statistical learning, AML algorithms are robust regarding the statistical properties of the data and are parameter-free.

The aim of the EU-funded ALMA project is to leverage AML properties to develop a new generation of interactive, human-centric machine learning systems. These systems are expected to reduce bias and prevent discrimination, remember what they know when they are taught something new, facilitate trust and reliability and integrate complex ethical constraints into human-artificial intelligence systems. Furthermore, they are expected to promote distributed, collaborative learning.

The ALMA project is structured into 8 interdependent Work Package (WP), each of them focusing on different aspects of the administration, research and development. According to their competencies and roles, partners have been identified as work package leaders. The competence of a WP leader includes quality control monitoring, risk assessment and contingency planning inside a specific work package. Every WP is composed of its own tasks and set of deliverables due in specified months.

Figure 1 presents a visual diagram of the interrelations of each WP, along with the sub-tasks assigned to each block. The size of each box is not related to the workload of each task, it serves solely aesthetic purposes.

WP1 is devoted to the administrative management and technical coordination of the project. Thus, it affects transversely all other work packages and tasks. It involves all legal, contractual, financial and administrative activities. Additionally, it is responsible for the progress monitoring, quality control, risk mitigation analysis and action plan design. WP1's leadership is divided between eProsima, in charge of the administrative management, and DFKI, accountable for the technical direction.

Similarly, the scope of WP8 encompasses all tasks of every work package. It is dedicated to the dissemination and exploitation of AML. The design of a communication strategy, planning of scientific workshops and the contribution and collaboration to standards are among its main tasks.

WP2 focuses on extending the research on foundations of AML. It is the theoretical base on top of which other work packages are built. It involves the generalization of AML systems, the comparison and coupling with traditional statistical machine learning systems, the interaction between humans and the AML learning method and, lastly, the collective learning of multiple AML systems.

WP6, on the other hand, serves as the middleware for efficient data sharing of AML-DL compliant data. The main outcome of this WP is the AML-IP, which offers an innovative framework capable of accommodating the distinct AML components and managing the intercommunication between them over LAN or the Internet. The AML-IP developed in the WP6 integrates all tasks of work packages 2, 3, 4 and 5. It is oriented to open-source tools and edge and cloud environments. WP6 also aims to extend the AML-IP to work with robotics and constrained device platforms.

However, AML-IP does not only provide a new tool to operate AML components, but it also serves as a generic AI integrating platform. The idea behind this design is to ease the dissemination and distribution of the AML-IP by providing a new middleware to operate conventional AI models, instead of restricting its use to AML. This feature may serve as entrance for users to start operating with AML.

On top of the first three tasks of WP2 lay all activities developed in the WP5. The main objective of this work package is to design techniques to learn world models, ethics and culture. WP5 must face the challenge of combining data-driven learning with expert knowledge, interactively including human input to build up more complex models.

This process will be achieved alongside WP3, which focuses on designing a human-readable language and accompanying tools for facilitating the encoding of existing knowledge, constraints and task goals in the form required by AML. The last step of this work package is to develop an AML accelerator, defined as a high-performance engine for heterogeneous compute platforms, using the features of AML collective learning.

WP4 also uses collective learning features as its foundations, along with the Human-AML interaction. Its purpose is to develop new designs and evaluation methods for ensuring effective human interaction with intelligent systems. This work package studies the relationship between users and AML learning and reasoning systems to design a working and interactive interface, which will be tested and evaluated.

On top of all the work developed lays WP7. Its main objective is to create a testbed for demonstrating the Human-Machine interaction approach with AML. Each proposed use case aims to illustrate a different example of how humans can successfully interact with the AML Engine. These cases use image classification, intelligent supportive tools and higher-level cognition for domestic assistance as fields of action.

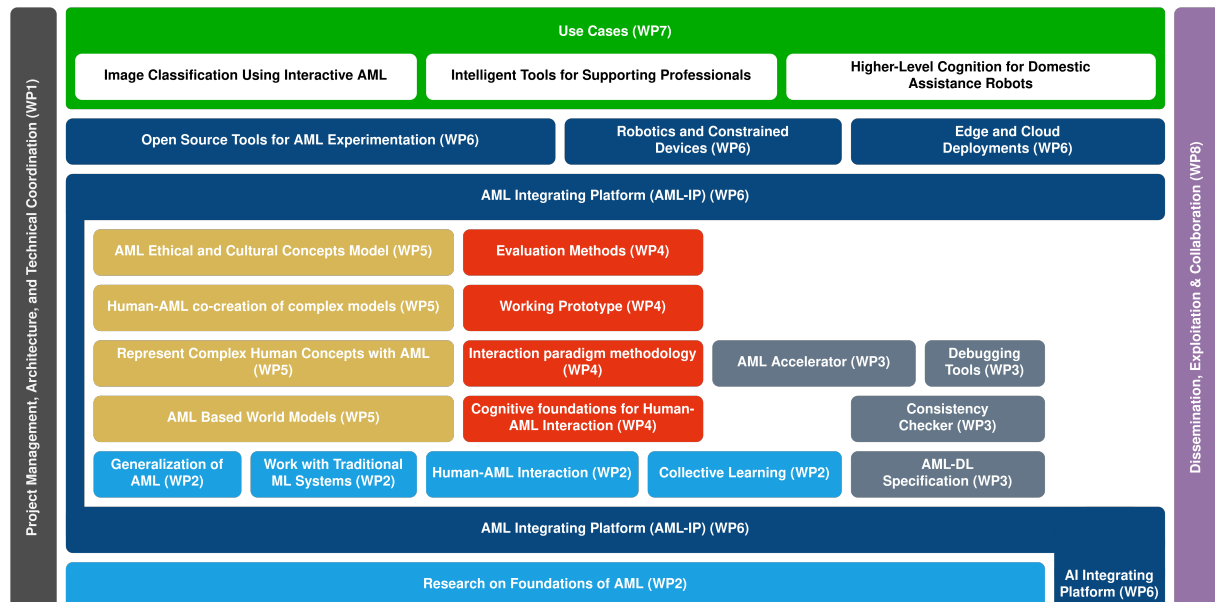


Figure 1: ALMA project overview and Work Package dependencies.

3 Algebraic Machine Learning

AML is a novel approach to ML grounded on Abstract Algebra and Model Theory, with no optimization procedure and with a lesser dependency on statistics [1, 2, 3]. One of its most appealing aspects, from an industrial point of view, is its capability for distributed computation, where the algorithm can theoretically scale up with no upper limit [4]. Thereby, the training dataset can be split into subsets, with the AML algorithm working on different subsets of the same problem in parallel and asynchronously across several machines. The final model can then be built by aggregating the individual contributions [4].

Note that parallel, multi-processor computing architectures are widely used in other ML methods, by running on a single machine with access to multiple processing units. This, however, does not leverage the whole potential for parallelism of AML. With a completely decentralised architecture, the AML algorithm is not limited to a single computer and can instead build models independently in several machines that are loosely coupled. These machines only need to submit their resulting model once their training phase is completed, additionally, they have the option to communicate knowledge asynchronously during training to boost learning [4].

The AML-IP aims to exploit this property, creating an environment for independent building blocks of the AML algorithm to communicate and share data and results.

3.1 AML environment

The AML environment is comprised of nodes of different kinds that communicate with each other. These nodes compute different types of data that are published and shared with other nodes in the AML ecosystem that request them.

One of the revolutionary aspects of AML is its great parallelization capability, allowing different kinds of nodes to share their data to distribute the workload and share their knowledge to boost each other's learning [4]. This implies that parallelization can be seen from two different angles:

- **Distributed workload:** The creation, processing and consumption of data is completely delocalized. Some nodes can create AML data that can later be processed by remote nodes in parallel executions.
- **Collaborative learning:** Different nodes can work on the same problem independently, using for instance different datasets. Sporadically, they can share the knowledge learnt with the other nodes, boosting their learning process and improving the accuracy of the model. Note that this also opens the door to the use of private datasets, as knowledge is shared but not the data itself.

3.1.1 Communication between AML nodes

The architecture on which this project focuses is based on the communication of different **AML nodes** through the Internet, freely distributing data and knowledge among them. The main points to be taken into account in order to properly understand AML-IP are:

- **Distributed architecture:** There is no main access point to the network, nor is there a centralised server to provide communication. This means Peer to Peer (P2P) communication is required between different AML nodes.
- **Data Distribution Service (DDS):** DDS will be the main communication protocol between nodes of the network.
- **World Wide Web (WWW):** This architecture is deployed over the Internet, allowing anyone with internet access to be part of the network.

Figure 2 presents a simple example of an AML execution. There are several nodes along the network that work under a DDS topic. Each node can publish under these DDS topics and/or subscribe to them, communicating with all other nodes within the AML network working on these same topics. Thus a P2P communication is achieved in which communication is not centralised.

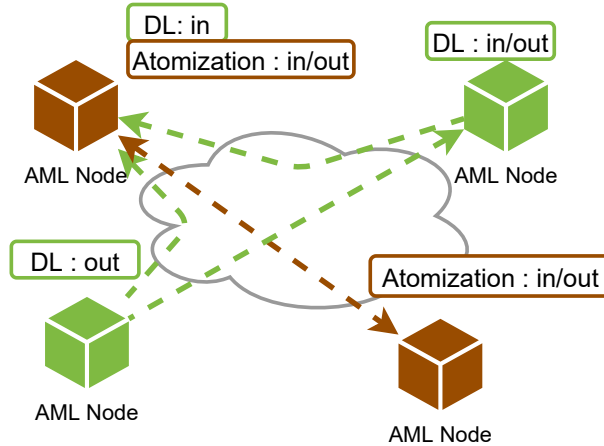


Figure 2: AML Nodes communicating with each other.

In the scenario shown in Figure 1, two types of AML nodes are presented. The green nodes communicate through the DL topic. This topic stands for the AML-DL, the language used by all AML nodes to communicate with each other. Moreover, the brown nodes represent AML computational engines. They produce a different type of data known as atomization. This atomization data can be either the input or the output of the node. Therefore, these nodes can both publish this data type under the Atomization DDS topic and/or subscribe to it.

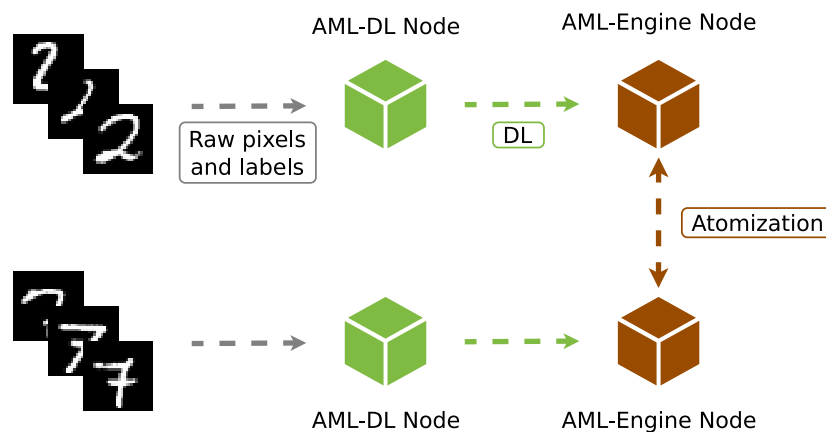


Figure 3: AML Nodes communicating with each other (MNIST dataset).

Figure 3 describes a network that learns how to identify handwritten numbers, for instance, those in

MNIST¹. Raw data is first passed to one or many AML-DL nodes, that interpret and publish it to the network using the DL topic. This data is then consumed by AML-Engine nodes that begin the learning process and also publish to and consume from the Atomization topic. To illustrate how data can be distributed, this example assumes that one single AML-Engine node only consumes samples corresponding all of them to the same label in the training data set, that is, they only learn to identify one single number. For instance, all samples labelled as “2” will be consumed by a single AML-Engine node and only by it. At the end of the training process, each AML-Engine node has a model able to identify one single number. These models can then be combined into one to be able to identify all 10 digits.

4 Scientific Contributions

The AML project has led to several publications, some of which are under peer review. Those publications follow different lines of research. In this section, we go briefly over some of this research.

Scientific contributions include core advancements in the understanding of atomized semilattices both in the finite [5] and infinite case [6]. Concepts and methods for the semantic embedding of goals and data into semilattices were formally described in [7], where it was shown that for a problem embedded in a semilattice, every solution has a model atomized by an irreducible subset of the non-redundant atoms of the freest model of the embedding.

A mathematical method and computational tools to embed real numbers into continuous semilattices were also developed. This advancement expanded the scope of AML, enabling its use with real-number inputs and outputs, thus facilitating work on numerical data, including regression tasks.

In a final report [8], we show that AML can perform image classification as well as the best-optimized MLPs, without using validation data. In this report, we also demonstrate that AML can solve hard problems such as Hamiltonian cycle finding. Furthermore, this paper presents a theorem stating that if there are hidden rules in the data, AML will find them. The report also includes a thorough study of the Sparse Crossing algorithm, pseudocode and a public open-source implementation. We hope this paper, which contains experimental data as well as 39 theorems, will definitively establish AML as a new method in the field of machine learning. The paper has been submitted to Nature Machine Intelligence.

Another line of research focused on applications and the human-interpretability of models. This research has involved extensive collaboration between consortium partners. For example [9] is related to expressive gesture typing while using a swipe keyboard focusing on human-centered explainability and explainable AI in general. The balancing of accuracy and complexity of semilattices was also studied in [10], where they were deemed more expressive than tree representations while also being more capable of balancing accuracy and complexity. Algebraic models were also analysed in the context of loopholes in [11] and in the context of cognitive load theory in [12]. Regarding ethics, contributions include exploring the technical opportunities and fundamental challenges of hybrid machine learning models (such as AML) as studied in [13]. Overall, these contributions provide some novel examples of how complex sociocultural concepts can be represented in interactive human-centric machine learning systems generally and with AML in particular.

¹The MNIST Database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>

5 AML Description Language

The goals of AML-DL are to give the user the capability to describe complex algebraic semantic embeddings in a simple and concise way, but also to establish a methodology to build embeddings for real-life problems. A small set of commands allows for the expression of large algebraic embeddings with brief AML-DL scripts. By using nested iterator commands a single AML-DL statement can translate into a large set of relations. This has already proven extremely useful in practice.

Although AML-DL may never be able to fully remove the need for a user with experience, there are primitives and methodologies that are integrated into AML-DL to make the embedding process more intuitive and streamlined. The goal of the language is to be flexible and adaptable so that the process is simple for the user. The embedding of a problem often requires its dissection into smaller parts. Some of these are common to many problems and these common parts are motifs for which specific AML-DL syntax could be provided, and so improving and expanding the language over time.

This language and the mathematical theory on which it is based are fully described in *Deliverable D3.4 - AML-DL Specification - Final M48*. Tooling has been created in *Deliverable D3.7 - AML-DL Interpreter - Final M48* and *Deliverable D3.9 - AML-DL Consistency Checker - Final M36*, to streamline the creation and exploration of AML-DL embeddings. In *Deliverable D3.11 - AML-DL Debugger - Final M48*, a tool that integrates an interpreter, a consistency checker, and an editor is presented.

6 AML Integrating Platform

The AML-IP is a communications framework in charge of data exchange between AML nodes through local or remote networks, which focuses on the easy deployment of AML nodes and the communication between them. Thus, AML-IP is not just a user-oriented application, but an intermediate library that makes the transport protocols transparent to the user. Therefore, it is thought to create a platform that communicates each node with those that are in the same AML network without requiring the user to know or understand the transport protocols behind each communication.

6.1 Dependencies

This section describes the AML-IP technical dependencies. All of them focus on the communication between nodes, the protocols used to support such communication and the tools used to handle the different types of data to be transmitted.

- ***eProsima Fast DDS***. Given the distributed nature of the AML ecosystem, the use of DDS is proposed to enable real-time communication between AML nodes. *eProsima Fast DDS* is a C++ implementation of the DDS specification and provides both an Application Programming Interface (API) and a communication protocol to deploy a DDS environment.

Moreover, the dynamic addition of new entities to the network is also covered by the DDS protocol, allowing the dynamic discovery of new instantiated DDS entities. The AML-IP framework will exploit the Discovery Server discovery mechanism described in [14].

Fast DDS also implements the TCP transport protocol for data transmission between DDS entities in Wide Area Networks (WANs). The purpose of the AML-IP is to implement a communications framework that uses the Transmission Control Protocol (TCP) communications protocol in a transparent manner for the user.

- ***eProxima DDS Router***². This is an end-user software application that enables the connection of distributed DDS networks. That is, DDS entities such as publishers and subscribers deployed in one geographic location and using a dedicated local network will be able to communicate with other DDS entities deployed in different geographic areas on their own dedicated local networks as if they were all on the same network.
- **Fast DDS Python Bindings**. This is an innovative development by eProxima that facilitates the utilization of *eProxima Fast DDS* within a Python environment. This is a very useful feature for the ALMA project, as most of the code produced for AML is written in Python, and Python will most likely be the chosen language for the main framework with which the end users will interact.

A more detailed explanation of these technologies can be found in *Section 2.5 Enabling technologies of Deliverable D6.4 - AML-IP implementation - Final M48*.

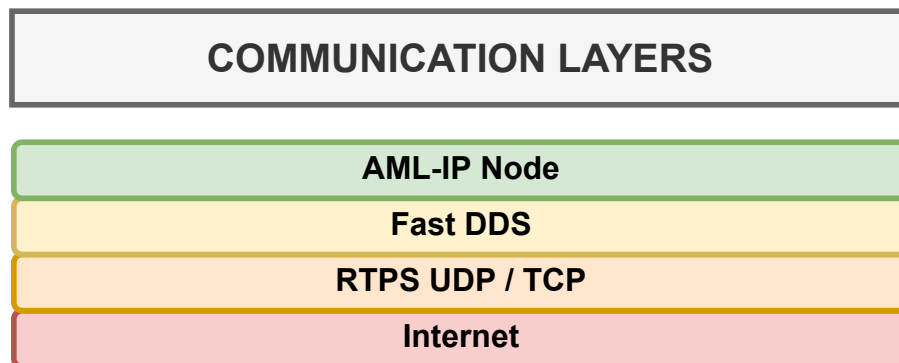
6.2 AML-IP Design

The chosen design is focused on the division of the different AML-IP components that constitute the network. In order to divide the workload, distribute the communication process and to create a scalable and widely usable architecture, the components of the AML-IP are divided into three categories, i.e. the AML-IP architecture consists of three types of nodes:

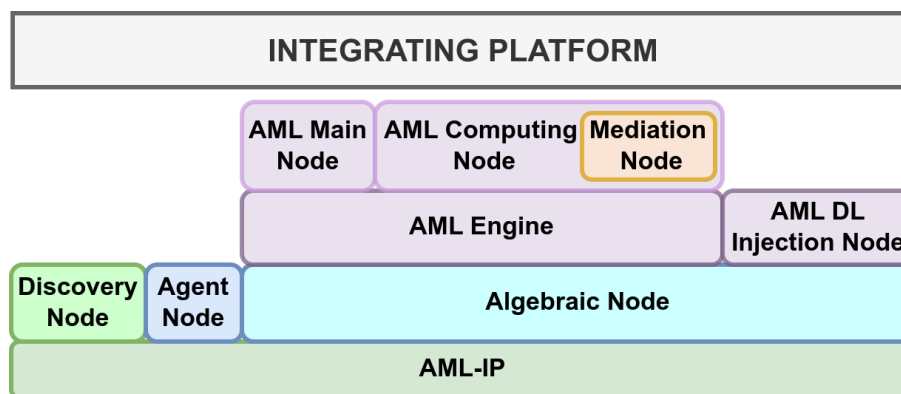
- **AML-IP Algebraic Node (AML-IP-Alg)**: These nodes represent each of the instances of the AML.
- **AML-IP Mediation Node (AML-IP-Mediation)**: These nodes handle data transformation between various formats. In particular, we focus on the preprocessing of AML-DL files and raw data that are translated to a data format that can be consumed by an AML Engine.
- **AML-IP Agent Node (AML-IP-Agent)**: This is the main transport component that manages the dynamic discovery of AML-IP Algebraic Nodes (AML-IP-Algs) deployed within a LAN and the discovery of other AML clusters over WAN.
- **AML-IP Discovery Node (AML-IP-D)**: These components will be disseminated along the network and made accessible through the Internet. They manage the WAN discovery process along the whole AML environment, handling the discovery of independent AML clusters and efficiently connecting the AML-IP-Alg that must connect.

The general architecture of the AML-IP is displayed in Figure 4. It shows how the AML-IP would integrate all transport and network elements (4a), as well as all types of AML-IP nodes and the hierarchy among them (4b).

²<https://github.com/eProxima/DDS-Router>



(a) AML-IP communication layer model.



(b) AML-IP nodes layer model.

Figure 4: AML-IP Architecture

Figure 5 shows a general example of an AML network and how the AML-IP will establish a communication method for the distributed nodes. The AML-IP Discovery Nodes (AML-IP-Ds) are network nodes that are distributed along the Internet and work as a *well-known address* in order to manage the WAN discovery. Other kinds of nodes are the AML-IP Agent Nodes (AML-IP-Agents), network nodes deployed by each user. They manage the connection between a LAN cluster and the rest of the network. The AML cluster is just a group of nodes inside the same LAN that can communicate with each other within the LAN to which they belong in a faster and more efficient way than through WAN.

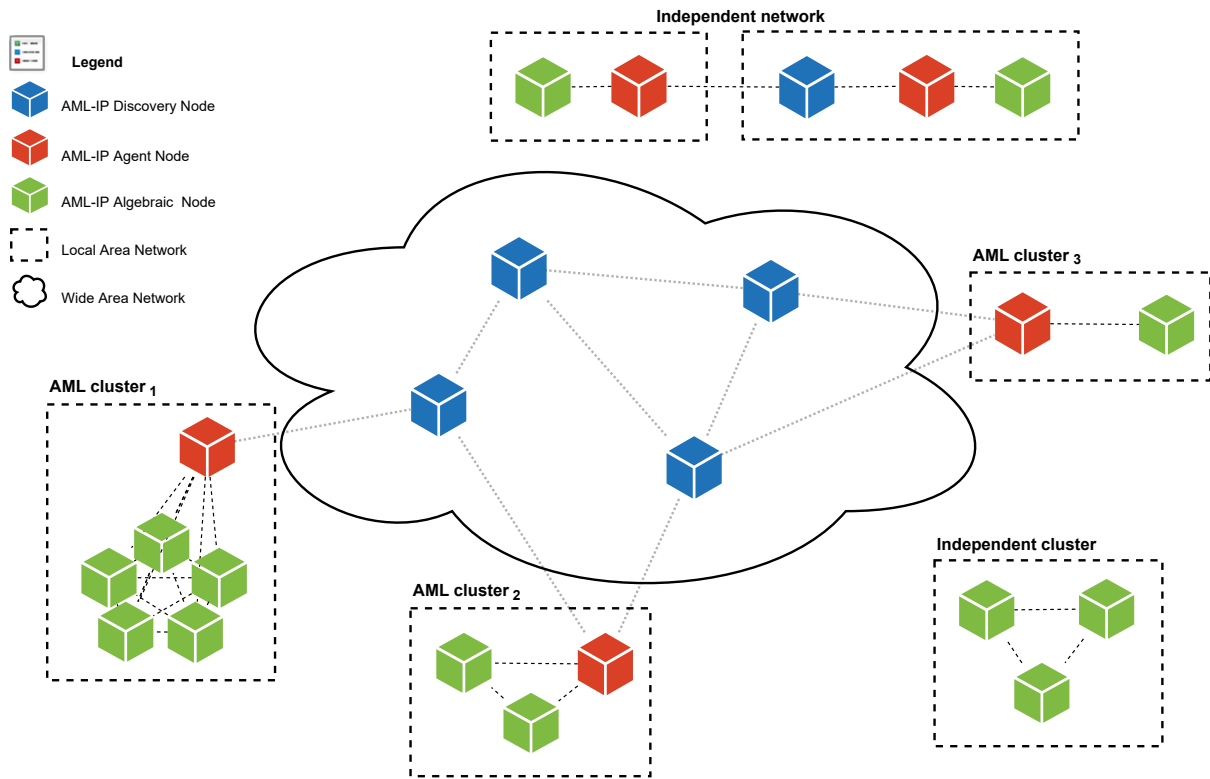


Figure 5: AML-IP example in LAN and WAN networks.

6.3 AML-IP functional operation

This section explains and exemplifies the communication process of all nodes within the AML-IP. The differences between the various stages of inter-node communication are also explained.

6.3.1 Phases of communication

The communication process in DDS is divided in two phases: the discovery phase and the data transfer phase:

- **Discovery phase:** The discovery phase establishes the mechanism by which new DDS entities are detected by other entities already operating in the network. New entities communicate their arrival in the network via messages to a multicast group and continue sending these messages to discover new entities or inform known entities of their current status. These messages are required to maintain the network state in a dynamic and distributed way.
- **AML data transfer phase:** This phase corresponds to the transfer of data required by the AML-IP-Alg to correctly execute their function.

6.3.2 AML-IP operational area networks

The networks where the AML-IP could operate are the Local Area Network (LAN), whether all the hosts are under the same switch or router and share a private network; or the Wide Area Network (WAN), where nodes are distributed along a wide network and usually behind different Network Address Translations (NATs).

In the figure 6 it is represented as an example of communication between different kinds of nodes deployed in LAN and WAN. The discovery and the data transfer phases depend on the network type. The nodes that are under the same LAN discover each other by *DDS Simple Discovery*, that is, by multicast, and communicate to each other independently of the rest of the internal or external network. In order to know the existence of external clusters with nodes that are required to interchange data, it must be deployed an AML-IP-D network that interconnects different LANs by distributing the discovery information. To manage the WAN discovery and data communication, an AML-IP-Agent must be deployed in every LAN and it centralizes the whole process to WAN communication. Two different AML-IP-Agent communicate with each other in P2P. They do not rely on the AML-IP-Ds that have connected them except to know the actual state or possible updates on the other network.

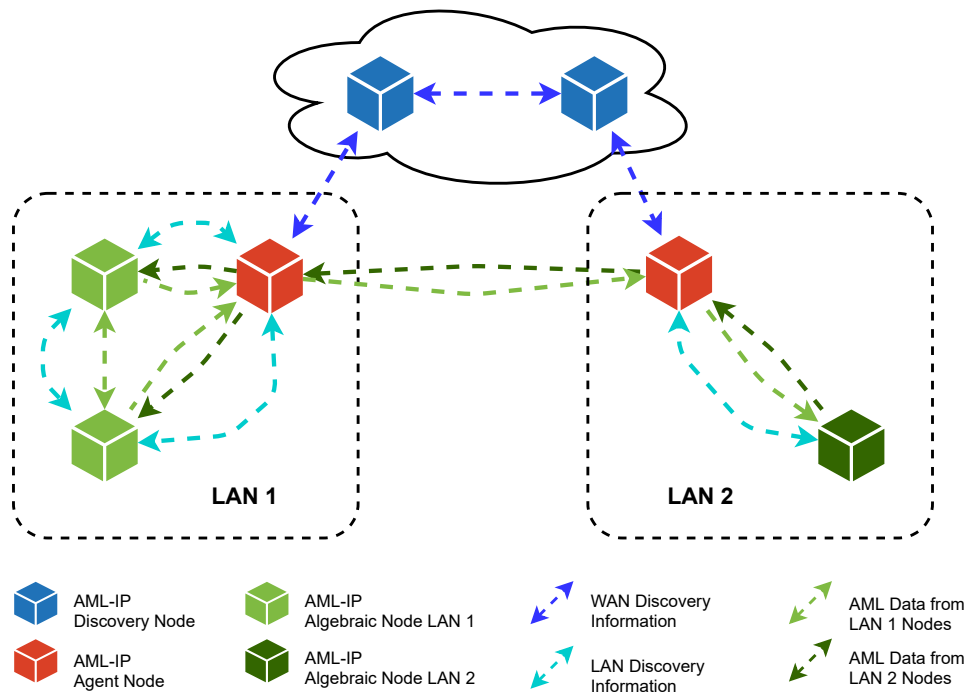


Figure 6: AML-IP minimum example to describe LAN and WAN communication.

A more detailed explanation of the functionality and design of each of the AML-IP components, as well as the detailed steps necessary to establish communication between them can be found in *Deliverable D6.4 - AML-IP implementation - Final M48*.

7 Current State

The AML algorithm can be executed in a non-distributed way by running the whole computation on a single machine. For this purpose, parallel, multi-processor computing architectures can be exploited in a similar fashion to how other ML methods deal with parallelism. This, however, would not leverage the whole potential for parallelism of AML. With a completely decentralised architecture as the one described in this manuscript, the AML algorithm is not limited to a single computer and can instead run independently in several machines.

The AML-IP³ is a platform designed to cover a wide range of actions in an AML network; it allows for the collaboration between different networks, keeping the privacy and independence of each node; it makes it possible to parallelize certain tasks between several local or remote nodes; it allows to merging several trained models to obtain a better one; and it also contemplates those use cases where multiple nodes may work jointly on the same task, by facilitating the communication and synchronization between nodes.

7.1 AML-IP workload distribution scenario

The so-called *workload distribution scenario* refers to those scenarios in AML where one computational complex task could be divided into subtasks, and these subtasks could be processed in parallel by different computational points, to finally obtain a joint result.

The workload distribution scenario directly applies to the training process performed in the AML-IP Algebraic Engine Node (AML-IP-Engine), the one in charge of training a model by adding new atomizations resulting from the processing of batches of data. This is a very expensive process that requires a high amount of memory and computational power. But it is also a parallelizable process, where some batches of data could be trained in parallel, and then the results could be merged to obtain a better model.

The main idea of this part of the AML-IP is to parallelize the computational effort in a balanced way between an arbitrary number of computing points. The work that is presented along this section refers to the process of creating new atomizations, generated by an AML-IP-Engine. It could be said that the scenario here exposed comprehends the detailed process of an AML-IP-Engine distributing its own workload among several computing points.

The nodes that take part in this sort of scenario are the following:

- **AML-IP-Main:** This kind of node functions as a central point for a *training* process. It creates *Jobs*, which could be seen as batches of data to be processed by the training algorithm. These jobs are distributed to several computing nodes (as any batch could be trained in parallel thanks to the AML), and their corresponding results are collected and merged back into the main node.
- **AML-IP-Computing:** This kind of node is in essence a computational point where a workload is released, so it can process a specific task on the data it has been given. This task corresponds to training over a batch of AML data and sending the pertinent result to the AML-IP-Main.

Figure 7 represents the previously described scenario. In this example, two AML-IP-Main work in the same problem, but with different datasets. Each of them has two different batches of training data to process, but since an initial preprocessing phase is needed, the two batches are not required to be executed at the exact same time. With this schema, it can be accomplished that, using three AML-IP-Computing, the workload of the two AML-IP-Main is distributed among them in an efficient way, so a task is resolved by a machine and other executions do not overlap. This is a specific use case, but any scenario with an arbitrary number of AML-IP-Main and AML-IP-Computing nodes is possible, and the workload would be *intelligently* divided between available and free nodes so the parallelization of complex tasks (as, in this case, the training phase) is maximized.

³<https://github.com/eProsima/AML-IP>

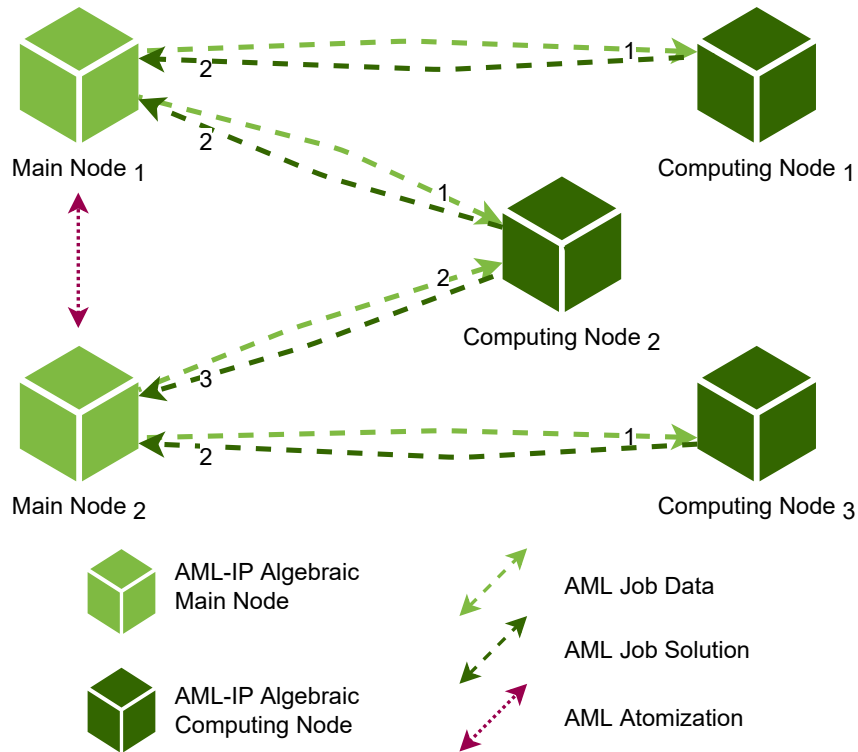


Figure 7: AML-IP workload distribution scenario.

7.2 AML-IP collaborative learning scenario

The so-called *collaborative learning scenario* refers to those scenarios in AML where nodes working on the same problem share their locally obtained models with each other. This sharing occurs without the need to expose the private datasets they were trained on, thus safeguarding sensitive information. The main objective is to collaboratively create a more complex and precise model by leveraging insights from diverse sources.

The main idea of this part of the AML-IP is to parallelize the computational effort in a balanced way between an arbitrary number of nodes. The work that is presented in this section refers to the process of sharing models, generated by an AML-Engine. Essentially, this scenario outlines how an AML-Engine disseminates its model to various computing points within the network.

In this scenario, two types of nodes are involved:

- **AML-IP Algebraic Model Manager Receiver Node (AML-IP-MMReceiver):** This kind of node performs the active (client) role. It receives *statistics* regarding available models and sends *requests* if it is interested in a particular one. Subsequently, it waits for the requested model to be delivered.
- **AML-IP Algebraic Model Manager Sender Node (AML-IP-MMSender):** This kind of node plays the passive (server) role. It shares *statistics* about the models it manages and awaits requests from other nodes. When a request is received, it responds by sending the corresponding model back as a reply to the requesting node.

Figure 8 illustrates the previously described scenario. In this example, an AML-IP-MMReceiver node and

an AML-IP-MMSender work on the same problem but with different datasets. Despite having distinct datasets, the AML-IP-MMReceiver node can learn from the model generated by the AML-IP-MMSender node without requiring direct access to its dataset. As a result, the AML-IP-MMReceiver node obtains an accurate model.

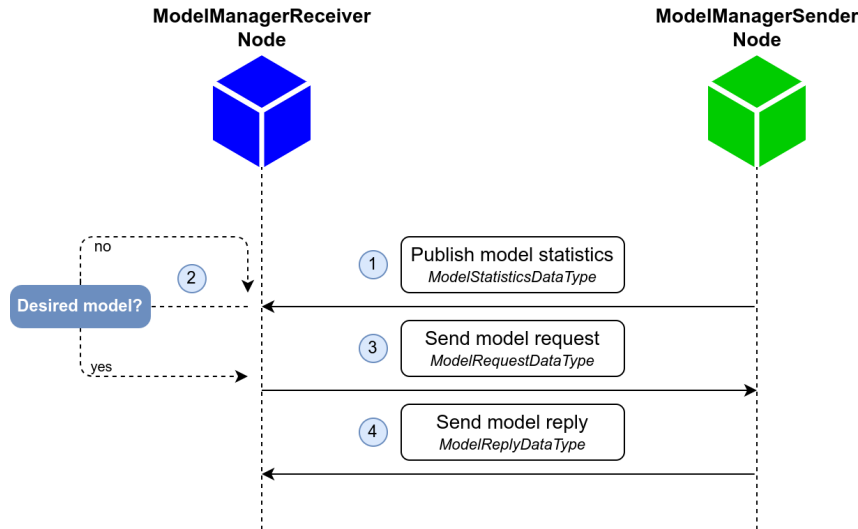


Figure 8: AML-IP collaborative learning scenario.

7.3 AML-IP distributed inference scenario

The so-called *distributed inference scenario* in AML involves the parallel processing of complex data by multiple computational points to obtain their inferences. In this context, large and computationally intensive tasks are divided and executed simultaneously across various AML-IP Algebraic Inference Node (AML-IP-Inference) nodes. This approach allows for faster and more efficient processing, leading to the ultimate inference or outcome.

The nodes involved in this sort of scenario are the following:

- **AML-IP Algebraic Edge Node (AML-IP-Edge)**: This type of node is responsible for sending data to other nodes within the network and then waiting for the inference to arrive. The AML-IP-Edge node initiates the data transmission and is involved in the data-sharing phase of the distributed inference process.
- **AML-IP-Inference**: This kind of node waits for data to be received from other nodes. Once it receives the necessary data, the node proceeds to calculate the inference, which represents the output or solution to the problem being addressed. After generating the inference, it sends the result back to the requesting node.

Figure 9 illustrates the previously described scenario. In this example, an AML-IP-Engine node and an AML-IP-Inference work on the same data but with different information. The AML-IP-Engine node can obtain the inference from the model of the AML-IP-Inference node without needing direct access to the underlying model.

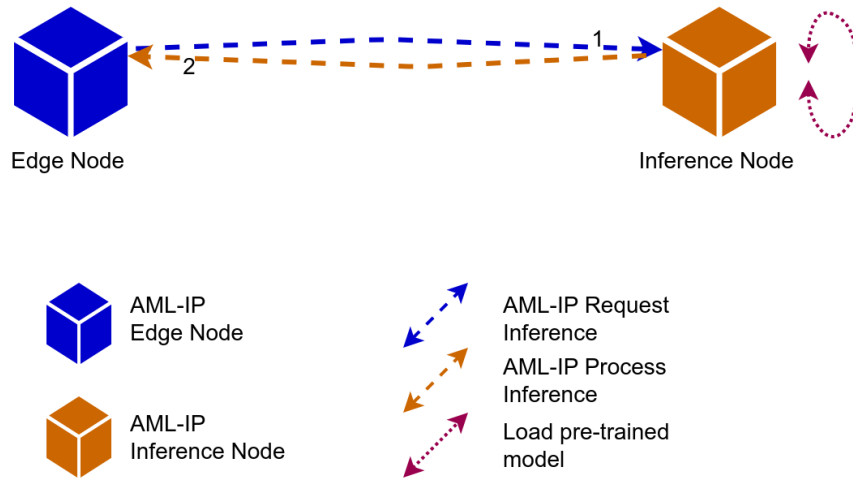


Figure 9: AML-IP distributed inference scenario.

7.4 AML-IP monitor network state scenario

The so-called *monitor network state scenario* pertains to specific instances within the domain of AML where the monitoring process involves activities encompassing comprehension, analysis, and troubleshooting of an AML network.

This scenario involves every node, as all of them publish the Status information. However, the only Node Kind properly belonging to this scenario is the following:

- **AML-IP Algebraic Status Node (AML-IP-Status):** This kind of node requires a few interactions with the user once it is running. User must start and stop this node as desired using methods *process_status_async* and *stop_processing*. Also, the user must yield a callback function that will be executed with every Status message received.

Figure 10 illustrates the previously described scenario. In this example, an AML-IP-Status node receives the status of an AML-IP-Engine node, an AML-IP-Inference node, that is communicating with the previous one, a AML-IP-MMSender node, and a AML-IP-MMReceiver node, which is communicating with the AML-IP-MMSender. As a result, the AML-IP-Status node obtains an accurate description of the AML network endpoints.

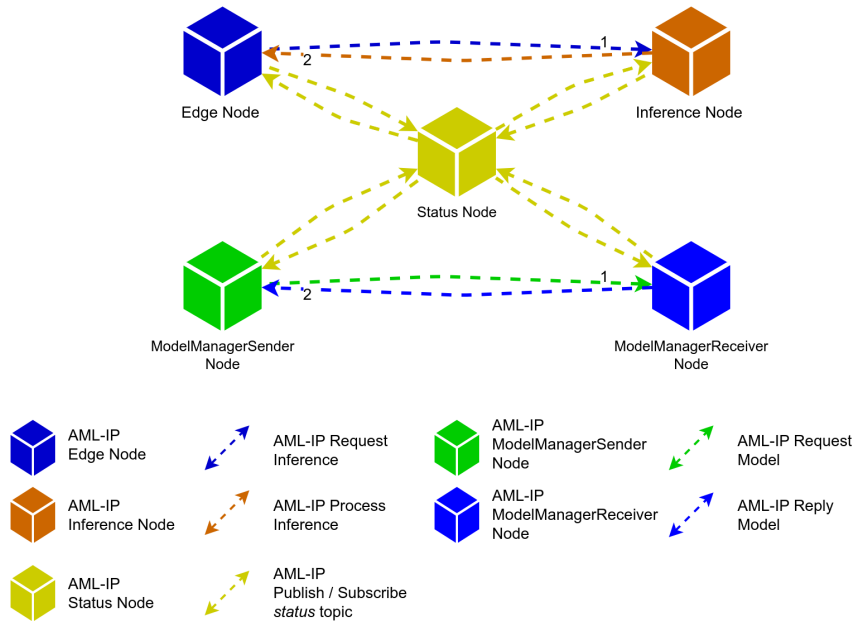


Figure 10: AML-IP monitor network state scenario.

7.5 Context Broker Integration

To integrate the FIWARE Context Broker with AML-IP, a Fiware Node was developed to manage inference data and facilitate interaction with the Context Broker. This node is responsible for handling inference requests by reading data from a context broker entity, requesting inference from an Inference Node, and transmitting the results back to the Context Broker. To streamline this process, the Fiware Node also incorporates an Edge Node to facilitate inference requests efficiently.

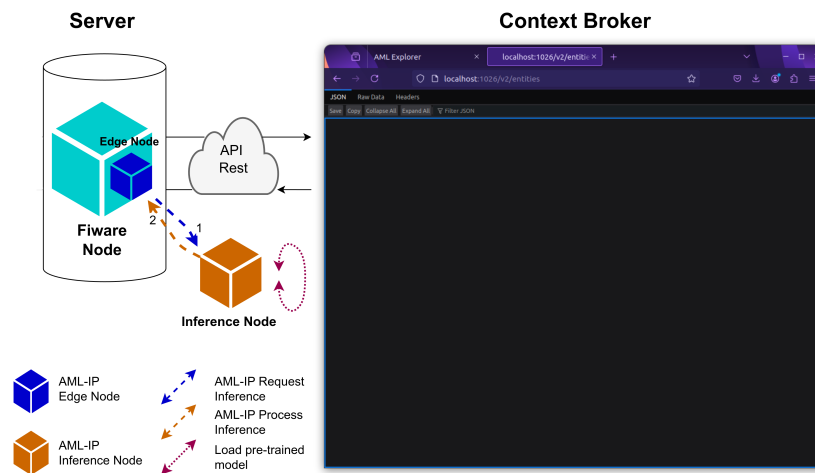


Figure 11: Fiware Node interaction with the Context Broker

For a comprehensive understanding of the solution devised for the specified scenarios consult *Deliverable D6.4 - AML-IP implementation - Final M48*. This document provides an in-depth exploration of the

proposed approach, encompassing details such as the specification of communication protocols like the (MultiService Communication Protocol (MSCP)) developed to handle the data exchange between these new kinds of nodes. Furthermore, it includes demonstrations showcasing practical cases.

8 AML-IP Use Cases

8.1 Training MNIST

The MNIST database is well known for providing images of handwritten digits, consisting of a training set of 60,000 examples, of which 10,000 are designated for testing, and the remainder for training. Derived from the NIST database, the MNIST digits have been size normalised and centred in a fixed-size image.

Training Workflow To train an MNIST model using AML-IP, a Workload Distribution scenario and a Collaborative Learning Scenario are employed, where different nodes handle distinct tasks in the training pipeline. This setup ensures efficient model distribution, training, and monitoring.

1. The **AML-IP-Main Node** initiates a request to obtain a model by distributing the MNIST training dataset.
2. The **AML-IP-Computing Node** processes the request, performs model training and returns the trained model to the AML-IP-Main Node.
3. The **AML-IP-MMSender Node**, which operates within the same process as the AML-IP-Main Node, publishes statistics associated with the trained model.
4. The **AML-IP-MMReceiver Node** listens for these statistics and requests the model based on the received information.
5. The **AML-IP-MMSender Node** responds to the request by providing the trained model.

Inference Workflow Once the MNIST model is trained, the inference is performed in a **Distributed Inference** scenario:

- The **Edge Node** sends data to an **Inference Node**, requesting a prediction.
- The **Inference Node** processes the input, computes the inference and returns the result to the requester.

8.2 AML-Dashboard

The **AML Dashboard** emerges as a user-friendly platform for training, evaluating, and applying AML models across several datasets.

The primary goal of the AML Dashboard is to enable users to:

- **Collect and Manage Data:** Record personal gesture data or load existing datasets.
- **Train Models:** Train classifiers using the AML Engine.
- **Evaluate Performance:** Compare AML models with other machine learning approaches.

The Dashboard is structured into various interfaces, each tailored to specific stages of the AML workflow.

Data Collection The **Data Collection** tab allows users to efficiently create, load, and manage datasets. Users can record gesture data, load standard image classification datasets, or upload custom datasets. A dataset browser enables easy navigation, offering tools to explore, modify, or delete images and manage class labels.

Training The **Training** page is where users can configure and monitor the training process of the AML model. It provides options to set training parameters, track the training progress, and compare the performance of the AML model with other classifiers. Additionally, users can train both the AML model and a neural network model for future comparison. Training an AML model follows a **Workload Distribution** scenario, requiring an active **Main Node** and at least one active **Computing Node** to efficiently handle the process.

Fetching This interface simplifies the retrieval of AML statistics and trained models. Users can search for specific statistics or request AML models with just a few clicks. It operates within a Collaborative Learning scenario, utilizing a Model Manager Sender and a Model Manager Receiver to facilitate model and statistics sharing.

Explore Models A confusion matrix is provided based on the trained model. Users must actively trigger the update of the matrix, facilitating a clear before-and-after comparison. This process follows a Distributed Inference Scenario, utilizing an Edge Node to send data and an Inference Node to process and return the results.

Real Time Pattern Exploration After training the models, users can move to the **Real Time Pattern Exploration** interface. This part of the AML Dashboard allows users to explore music patterns using the self-trained models. This feature is only available when the dataset contains captured gesture data. This process also follows a Distributed Inference Scenario.

Context Broker Interaction This tab facilitates integration with the Context Broker, allowing users to upload images, perform inference tasks, and receive predictions from trained AML models. This uses a Fiware Node to communicate with Context Broker and handle inference data.

AML-IP Nodes Management This interface facilitates the management of the AML-IP nodes within the network. Users can switch between tabs to create and destroy agent, computing, inference, and sender nodes.

Status The **Status** tab offers a comprehensive view of all active AML-IP nodes, displaying their ID, type, and operational status. It operates within a Monitor Network State scenario, utilizing a Status Node to collect and consolidate status information from all other nodes, ensuring efficient system monitoring and management.

The AML Dashboard can be installed using the detailed steps provided in the AML-Tools Documentation⁴ as well as diving into more comprehensive tutorials to get started.

⁴<https://aml-tools.readthedocs.io/en/latest/rst/installation/linux/amldashboard.html#amldashboard-linux>

8.3 AML-Toolkit

AML Toolkit⁵ is a set of software tools that allows users to develop, train, and deploy AML models. These tools are designed to be used by researchers, developers, and industry professionals who want to implement AML models in their applications. AML Toolkit aims to provide users, developers, and researchers with the documentation, guides, and examples needed to start working with AML.

These tools cover the entire process of developing AML applications, from data collection and preprocessing, to generating embeddings for problem description, validating those embeddings, training the model using an AML engine, performing inference using the resulted AML model, and validating the results.

To access the AML Toolkit Documentation, please visit: <https://aml-tools.readthedocs.io/>.

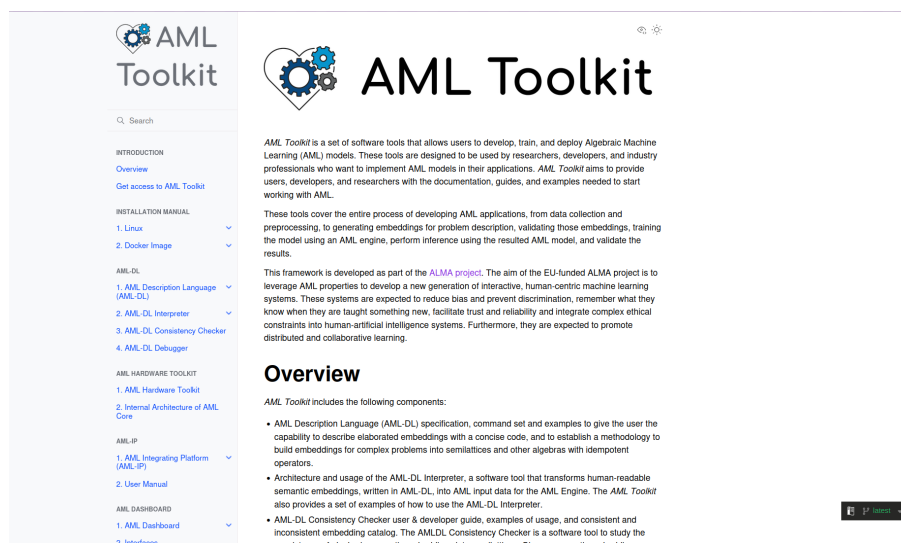


Figure 12: AML Toolkit Documentation

9 Innovation Management

According to *Section 3.2.2 Innovation Management* of the description of work included in the Grant Agreement (GA) 952091 of the ALMA project, *"the partners in the Consortium undertake to provide a maximum level of transparency in terms of the available knowledge for the purposes of effective and efficient project execution"*.

The measures taken to ensure maximum awareness of the Project Officer (PO) of the available knowledge and internal communication within the ALMA Consortium are:

- *"Setting up and maintaining a project web page that has a public section to communicate, for example, ongoing activities and project progress, a list of recent publications and events. As described in Section 3 Open access policy of ALMA of Deliverable D8.11 - Data Management Plan - Revised"*

⁵<https://aml-tools.readthedocs.io/en/latest/rst/formalia/titlepage.html>

M48, all information on the public knowledge produced and the planned schedule of dissemination activities of the ALMA project has been published on the ALMA website⁶.

- *"A private section to facilitate communication within the Consortium, e.g. by means of mailing lists and a document management section."* This measure is described in *Section 5.2 Making data openly accessible* of *Deliverable D8.11 - Data Management Plan - Revised M48*.
- *"Setting up and maintaining a project repository for publications, software, etc."* As in the previous point, this measure is described in *Section 5.2 Making data openly accessible*, *Section 5.4 Increase data re-use*, and *Section 6 Allocation of resources* of *Deliverable D8.11 - Data Management Plan - Revised M48*.

With regard to the dissemination and exploitation of the knowledge generated in the ALMA project, the ALMA Consortium has complied with the directives outlined in *Section 3 Open access policy of ALMA* of *Deliverable D8.11 - Data Management Plan - Revised M48*. As stated in this deliverable:

all deliverables already defined and the data generated will be publicly available, except for the ethical reports, whose distribution will be private and internal to ALMA consortium members. The research data will be uploaded to a public domain repository, and/or published in scientific journals, thus making it openly accessible. (...)

However, for each of the datasets and knowledge generated, the ALMA consortium will analyze the potential conflicts of openly publishing the results versus the commercialization and application of IPR protection, thus deciding what information will be public and when it will be public. (...)

10 Conclusions

This deliverable has described the general architecture of the AML environment, the AML nodes that comprise it, as well as the general design of the AML-IP and the elements that constitute it, the AML-DL definition and the scientific contributions made to AML.

AML-DL specification, command set and examples to give the user the capability to describe elaborated embeddings with a concise code, and to establish a methodology to build embeddings for complex problems into semilattices and other algebras with idempotent operators.

While AML-DL cannot entirely eliminate the need for experienced users, its built-in primitives and methodologies simplify and streamline the embedding process, making it more intuitive.

In parallel, several publications have been made during the development of this project, advancing both theoretical and applied aspects of AML. These contributions have deepened the understanding of atomized semilattices, explored semantic embeddings, and enhanced human-centred AI by balancing accuracy, complexity, and explainability. Ethical considerations in hybrid models further highlight AML's impact, demonstrating its potential in interpretable and interactive machine learning.

Complementing these efforts, AML-IP has been introduced as a communications framework designed to leverage the collaborative and distributed learning capabilities of AML. It enables the creation of a network of AML clusters that operate collaboratively across different geographical locations over the Internet, requiring minimal user configuration.

⁶<https://alma-ai.eu/>

This approach presents key challenges, such as the dynamic discovery of AML nodes working on the same problem and efficient communication across WANs. By addressing these challenges, AML-IP allows users tackling the same AML problem to seamlessly share their learning results without needing expertise in communication protocols.

AML-IP has been updated to version 0.3.0⁷, along with revised documentation and practical examples [15], streamlining its integration with other work packages in the ALMA project.

This framework is designed with a strong emphasis on network scalability and communication security, considering constraints such as data centralization at intermediate nodes and the high volume of data transmission required in distributed learning environments.

Finally, the innovation management plan is presented. This defines the guidelines to be followed in the publication of the ALMA project outcomes, as well as the measures taken to achieve the highest level of transparency of the work performed by the partners towards the PO.

⁷<https://github.com/eProxima/AML-IP/releases/tag/v0.3.0>

References

- [1] Fernando Martin-Maroto and Gonzalo G. de Polavieja. *Algebraic Machine Learning*. 2018. arXiv: 1803.05252.
- [2] Fernando Martin-Maroto and Gonzalo G. de Polavieja. *Finite Atomized Semilattices*. 2021. arXiv: arXiv:2102.08050.
- [3] Fernando Martin-Maroto and Gonzalo G. de Polavieja. *Semantic Embeddings in Semilattices (In preparation)*. 2021.
- [4] *patent US20190385087A1: Method for large-scale distributed machine learning using formal knowledge and training data*. 2018.
- [5] Fernando Martin-Maroto and Gonzalo G. de Polavieja. *Finite Atomized Semilattices*. 2021. arXiv: 2102.08050 [math.RA]. URL: <https://arxiv.org/abs/2102.08050>.
- [6] Fernando Martin-Maroto et al. *Infinite Atomized Semilattices*. 2023. arXiv: 2311.01389 [math.AC]. URL: <https://arxiv.org/abs/2311.01389>.
- [7] Fernando Martin-Maroto and Gonzalo G. de Polavieja. *Semantic Embeddings in Semilattices*. 2022. arXiv: 2205.12618 [cs.DM]. URL: <https://arxiv.org/abs/2205.12618>.
- [8] Fernando Martin-Maroto et al. Algebraic Machine Learning: Learning as computing an algebraic decomposition of a task. 2025. arXiv: 0000.00000.
- [9] Janin Koch and Vitor Fortes. “Combining Human-centred Explainability and Explainable AI”. In: *CHI 2022-Conference on Human Factors in Computing Systems*. 2022.
- [10] Stephen Fox and Antonio Ricciardo. “Better than Trees: Applying Semilattices to Balance the Accuracy and Complexity of Machine Learning Models”. In: *Machine Learning and Knowledge Extraction* 7.1 (2025). ISSN: 2504-4990. DOI: 10.3390/make7010005. URL: <https://www.mdpi.com/2504-4990/7/1/5>.
- [11] Stephen Fox and Juan G. Victores. “Safety of Human–Artificial Intelligence Systems: Applying Safety Science to Analyze Loopholes in Interactions between Human Organizations, Artificial Intelligence, and Individual People”. In: *Informatics* 11.2 (2024). ISSN: 2227-9709. DOI: 10.3390/informatics11020036. URL: <https://www.mdpi.com/2227-9709/11/2/36>.
- [12] Stephen Fox and Vitor Fortes Rey. “A Cognitive Load Theory (CLT) Analysis of Machine Learning Explainability, Transparency, Interpretability, and Shared Interpretability”. In: *Machine Learning and Knowledge Extraction* 6.3 (2024), pp. 1494–1509. ISSN: 2504-4990. DOI: 10.3390/make6030071. URL: <https://www.mdpi.com/2504-4990/6/3/71>.
- [13] Stephen Fox and Vitor Fortes Rey. “Representing Human Ethical Requirements in Hybrid Machine Learning Models: Technical Opportunities and Fundamental Challenges”. In: *Machine Learning and Knowledge Extraction* 6.1 (2024), pp. 580–592. ISSN: 2504-4990. DOI: 10.3390/make6010027. URL: <https://www.mdpi.com/2504-4990/6/1/27>.
- [14] *eProxima Fast DDS Documentation - What is DDS?* Version 2.3.1. Proyectos y Sistemas de Mantenimiento SL (eProxima), 2021. URL: https://fast-dds.docs.eprosima.com/en/v2.3.1/fastdds/getting_started/definitions.html.
- [15] *eProxima AML-IP Documentation*. Version latest. eProxima, 2022. URL: <https://aml-ip.readthedocs.io/en/latest/>.