



Human-centric algebraic machine learning

AML Complete Foundations- Revised

Deliverable D2.4

WP2 - Fundamentals of Interactive
Algebraic Machine Learning



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



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: **D2.4**
 Title: **AML Complete Foundations-Revised**
 Month: 54
 Work Package: WP2 - Fundamentals of Interactive Algebraic Machine Learning
 Work Package leader: Champalimaud Foundation
 Deliverable leader: Champalimaud Foundation
 Deliverable type: Report (R)
 Dissemination level: Public (PU)
 Date of submission: 2025-02-28
 Version: v1.1
 Status: Finished

Version history

Version	Date	Responsible	Author/Reviewer	Comments
v1.0	2025-02-25	CF	Fernando Martin-Maroto and Gonzalo G. de Polavieja	Draft and arXiv manuscript
v1.1	2025-02-27	CF	David Méndez	Reviewed
v2.0	2025-05-01	CF	Gonzalo G. de Polavieja	Revised
v2.1	2025-05-08	CF	Fernando Martin-Maroto	Revised

Executive summary

This line of deliverables is devoted to establishing the mathematical foundations of [Algebraic Machine Learning \(AML\)](#).

This is *Deliverable D2.4 - AML Complete Foundations-Revised* of the ALMA project, consisting of the final and revised foundations of AML. Deliverables D.2.1, D.2.2, and D.2.3 presented the mathematical foundations of AML. This revised version focuses on applying those foundations to machine learning. As a result of the work in this deliverable, we have written a manuscript available on arXiv: <https://arxiv.org/abs/2502.19944>.

The deliverable includes: (1) a description of AML as a method within ML; (2) an explanation of how AML generalizes to test data; (3) a performance comparison with neural networks; and (4) an illustration of how AML can also be applied to problems without examples. A more technical discussion is provided in the Supplementary Materials of <https://arxiv.org/abs/2502.19944>, which also includes an open-source implementation of an AML Python/C engine, available on [GitHub](#).

Figure 1 provides a schematic representation of the AML pipeline. We begin with axiomatization, wherein the problem, defined by data, goals, and prior knowledge, is encoded as a set of axioms in an algebra. Users must perform this axiomatization for new problems or simply add their data into existing axiomatizations that we provide, such as data for classification tasks or specific nodes to compute a Hamiltonian cycle. Next, we apply a procedure called Full Crossing to derive a model from the axioms. The resulting model possesses two key characteristics. First, it is the freest model, meaning only the axioms and their logical consequences hold true within it. Second, it is represented as a subdirect decomposition $\{\phi_1, \phi_2, \dots\}$, a method from Abstract Algebra [1], through which we identify the fundamental building blocks, or atoms, of the problem. These atoms emerge organically as data arrives, without any need for hyperparameter tuning or a validation dataset. We show which specific subsets of these atoms constitute generalizing models. In practical applications, users employ Sparse Crossing, a sparse version of Full Crossing that directly identifies generalizing subsets from the axioms.

In this deliverable, we describe each of these steps.

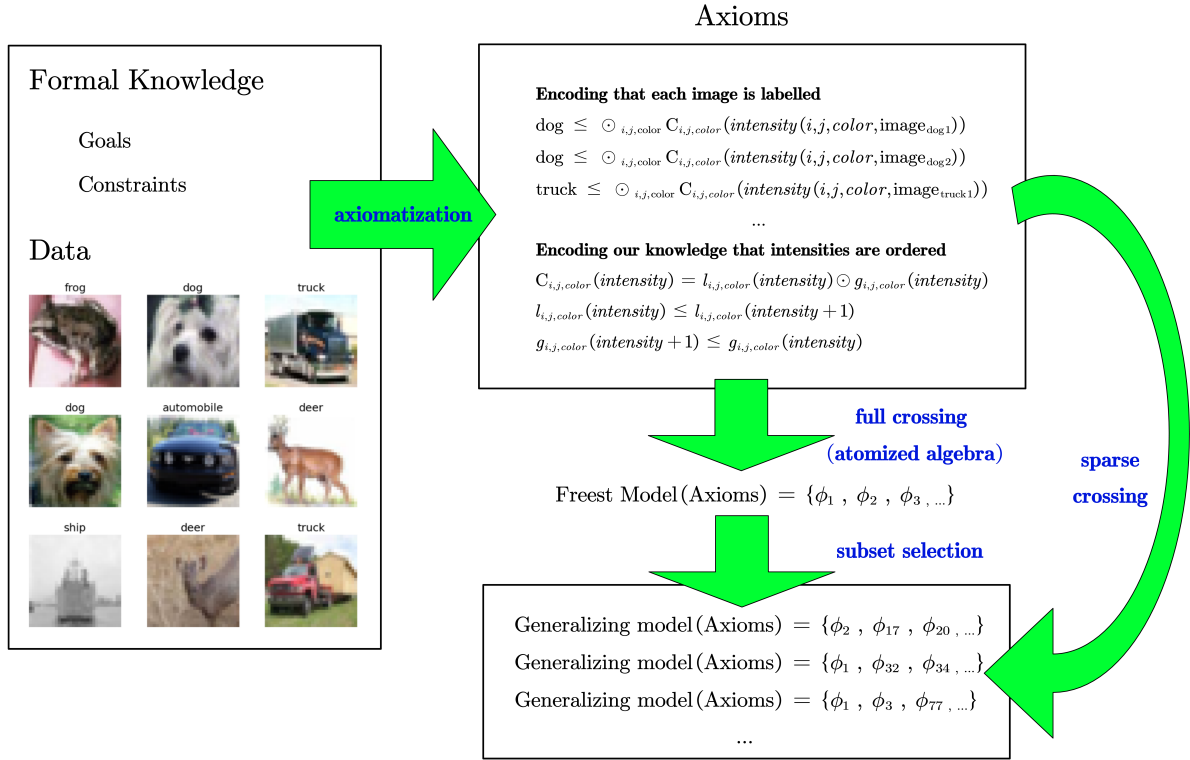


Figure 1: **Schematic representation of the Algebraic Machine Learning pipeline.** The process begins with axiomatization, where the problem, defined by data, goals, and prior knowledge, is encoded as a set of axioms. Then, we apply the Full Crossing procedure to obtain a specific model of the axioms, the freest model, a model in which the only true statements are the axioms and their logical consequences. Furthermore, the model is given explicitly as a subdirect product, expressed as basic atomic components (the atoms). Generalizing models are obtained by selecting certain subsets of atoms that collectively satisfy the axioms. In practical implementations, computing all atoms of the freest model is unnecessary; instead, a sparse variant of the Full Crossing procedure is used to directly calculate generalizing subsets of atoms.

Encoding a task as axioms

We encode a Machine Learning task algebraically as axioms. Each axiom defines a relationship between two terms, asserting that term T_L is contained in or implied by term T_R when

$$T_R \odot T_L = T_R, \quad (1)$$

or, more compactly,

$$T_L \leq T_R. \quad (2)$$

A Machine Learning task thus consists of positive axioms, each of the form $T_L \leq T_R$, and negative axioms, each of the form $T_L \not\leq T_R$.

As an example, consider encoding the property of binary sequences of length 4. Assign a constant p for the property and two constants per position to represent digits 0 and 1, totaling 8 constants. For instance, the constant c_{31} might indicate that the third position is 1. To express that the sequence 0100 belongs to class p , we write an axiom $T_L \leq T_R$ with $T_L = p$ and $T_R = c_{10} \odot c_{21} \odot c_{30} \odot c_{40}$. The task is encoded as a set of such axioms, one for each sequence in the class.

Atomized models

Once the task is encoded as axioms, the next step is to build an *atomized model*. Each axiom $T_L \leq T_R$ is satisfied if the atoms linked to constants in T_L form a subset of the atoms linked to constants in T_R .

Training an atomized model involves synthesizing a set of atoms that fulfill all positive axioms and violate all negative axioms. An atom ϕ is uniquely defined by the constants it is linked to, conveniently denoted as $\phi[c_3, c_4]$.

An atomized model is characterized entirely by its atoms, for example as

$$M = \{\phi[c_1, c_2, c_3], \phi[c_2, c_5], \phi[c_1, c_6], \phi[c_3], \phi[c_3, c_4], \phi[c_2, c_3, c_5]\}. \quad (3)$$

We compute the freest model using a procedure called Full Crossing, which we described in D2.1 and in <https://arxiv.org/abs/2502.19944>. To build intuition about the freest model, consider the task of characterizing a property p for the set of 3,375 black-and-white 4×4 images. The first column of each image is entirely black, while the remaining three columns contain pixels that are either black or white, with no column being completely black. **Figure 2a** displays 16 images meeting this criterion.

For this problem, we use 32 constants representing the black or white states of the 16 pixels, plus one constant for property p , totaling 33 constants. The initial model is the freest semilattice atomized by $\phi[c_{11w}], \phi[c_{11b}], \phi[c_{12w}], \dots, \phi[c_{44w}], \phi[c_{44b}], \phi[c_p]$, with indices of the form ijw or ijb indicating the position in the image matrix and the pixel color as white (w) or black (b).

Examining the final model in **Figure 2c**, each atom, such as the last atom $\phi[p, c_{13w}, c_{23w}, c_{33w}, c_{43w}]$, is illustrated by an image with a gray background highlighting pixels at positions 13, 23, 33, and 43 in white, excluding the constant p from the plot. The final model contains 51 atoms. Among these, 32 atoms correspond to single-pixel color constants, which were present in the initial model before any crossings. There are also 4 atoms consisting of two constants each—property p and a constant representing a black pixel from the first column—capturing the invariant presence of the black first column. Additionally, there are 12 atoms with three constants each—property p and both black and white constants for each pixel outside the first column—representing the variability of these pixels. Lastly, there are 3 atoms with five constants each—property p plus all four white constants from one of the three last columns—indicating no complete black columns in these positions.

It is also informative to inspect an intermediate model early in the crossing sequence, for example

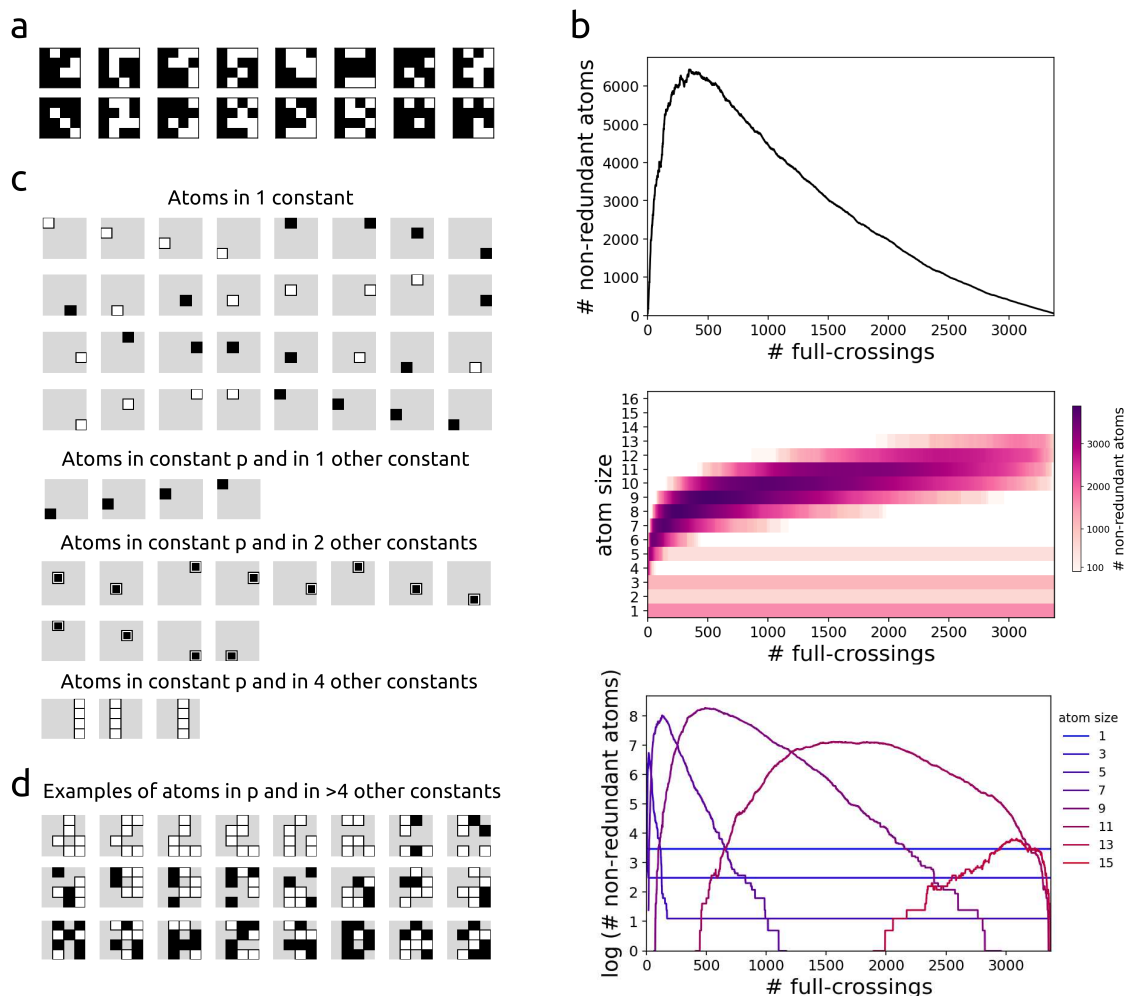


Figure 2: **Freest models using images with the only first column in black.** (a) 16 of the possible 3,375 images with only the first column in black. Training examples are of the form $p < T_i$ with T_i the term representing an image. (b) Top: Number of non-redundant atoms of the model obtained after a number of full-crossings. Middle: Number of non-redundant atoms of a given atom size for the model obtained after a number of full-crossings. Bottom: Same as Middle but represented by several curves, each for a different atom size. (c) Atoms of the final model. (d) Example of large atoms that are part of the models at intermediate number of full-crossings.

after 200 full crossings. This intermediate model already includes all atoms from the final model (**Figure 2c**) and contains larger atoms (examples shown in **Figure 2d**), which will ultimately be eliminated in subsequent crossings.

Atomized models give the underlying rule in the data

In the previous section, we considered the task of assigning a property p to a set of images, governed by the hidden rule that each image's first column is entirely black, and the remaining columns each contain at least one white pixel (**Figure 2**). The final freest model explicitly reveals this hidden rule through its non-redundant atoms, as shown in **Figure 2c**.

This finding is a general result. Let P be the set of axioms defining the hidden rules of a task, and let Q be the set of all axioms that logically follow from P , excluding P itself. It can be proven that the non-redundant atoms in the freest model derived from Q are identical to those in the freest model derived from P (see proof in <https://arxiv.org/abs/2502.19944>). The theorem thus says that, given enough task axioms (a substantial subset of Q), the freest model constructed from these axioms converges to the model representing the underlying rules (P).

Although exact convergence occurs in the limit, after the model is trained on all axioms in Q , the essential non-redundant atoms or an approximation to them typically emerge much earlier. In the previous example, the complete model required 3,375 crossings to finalize, yet its fundamental non-redundant atoms (**Figure 2c**) appear already within 200 crossings. Extracting these atoms at that earlier stage gives a generalizing model. The subsequent sections examine why early convergence to a subset of atoms that encode the task's underlying rules is a general phenomenon.

How atoms evolve during training

To understand how atoms evolve during training, consider the enforcement of positive task axioms $r_1, r_2, \dots, r_n$. **Figure 3** illustrates this for a simple case with $n = 3$. Initially, model N_0 does not satisfy any task axioms. The Full Crossing procedure is applied to the initial model N_0 to enforce axiom r_1 , producing model N_1 . Repeating this for each subsequent axiom generates a sequence of models: $N_1, N_2, \dots, N_n$.

For each atom ϕ in the final model N_n , there exists a corresponding *chain* of atoms $\lambda_0, \lambda_1, \dots, \lambda_n$, with each λ_i belonging to model N_i and $\lambda_n = \phi$ (illustrated in **Figure 3**, blue line). Within this chain, atoms may either change (marked in the figure as a red dot) or remain constant (green dot). A significant parameter for a final atom is $h(\phi)$, the number of successful matches the atom has had since its creation. In the example chain from **Figure 3**, we have $h(\phi_4^{(3)}) = 2$.

Consider extracting a subset of Z atoms from the final freest model N_n . Each atom ϕ_i in this subset has succeeded $h(\phi_i)$ times. The Probability of a False Negative (PFN) on the test set, meaning a positive test example classified incorrectly as negative, is approximated by:

$$\overline{\text{PFN}}(\phi_1, \dots, \phi_Z) \approx \sum_{i=1}^Z \frac{1}{h(\phi_i)}. \quad (4)$$

Thus, reducing test PFN involves selecting fewer atoms with higher training success ($h(\phi)$).

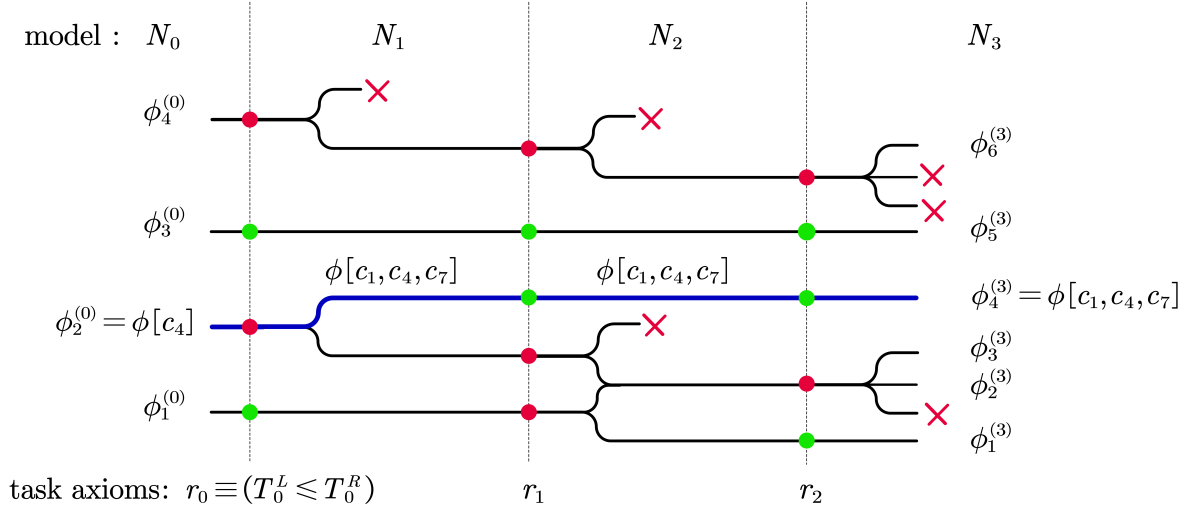


Figure 3: **Evolution of atoms during learning.** Starting from an initial model N_0 , enforcing axioms r_0, r_1 , and r_2 produces a sequence of models: N_0, N_1, N_2, N_3 . An atom ϕ in the final model N_3 can be traced back through atoms in each intermediate model (N_2, N_1, N_0), forming at least one chain of atoms $\lambda_i \in N_i$ with $\lambda_3 = \phi$. Throughout the chain, atoms change (red nodes) or remain unchanged (green nodes). Atoms marked with a red cross are redundant and can be discarded. The blue line highlights an chain from a final atom back to an initial atom. In this chain, there is one instance of atom growth ($g(\phi_4^{(3)}) = 1$), and the final atom has succeeded twice since its last growth ($h(\phi_4^{(3)}) = 2$).

The Probability of a False Positive (PFP), indicating negative test examples incorrectly classified as positive, is given by:

$$\overline{\text{PFP}}(\phi_1, \dots, \phi_Z) = \prod_{i=1}^Z \overline{\text{PFP}}(\phi_i). \quad (5)$$

Hence, selecting atoms effective in distinguishing negative examples in training reduces test PFP.

Equations 4 and 5 guide the extraction of generalizing models from the freest atomized model. Minimizing test PFN involves choosing fewer, highly successful atoms. Minimizing test PFP requires selecting atoms that effectively discriminate negative examples, exhibit low correlation among themselves, and collectively discriminate all negative training examples.

Applying this approach to the example in **Figure 2**, the "rule" atoms can be identified before 200 crossings. Typically, computing the full freest model is impractical due to its size. Instead, subsets of atoms are efficiently identified using the Sparse Crossing procedure, detailed in the Supplement of <https://arxiv.org/abs/2502.19944>.

Learning from examples

The way AML learns differs fundamentally from Neural Networks (NNs) (see **Figure 4**). In NNs, we must make numerous decisions about the model—such as its architecture, loss function, activation

functions, optimization method, and weight initialization. The validation dataset is used not only for early stopping but also to ensure that these choices yield high accuracy. If performance is insufficient, we typically revise and adjust these design decisions. In contrast, AML involves only an algebraic formulation (hyperparameter-free) of the problem, with no need to determine architectures or tune hyperparameters. As a result, AML does not require a validation dataset and is inherently resistant to overfitting.

	ALGEBRAIC MACHINE LEARNING	NEURAL NETWORKS
SETTING UP LEARNING SYSTEM	1. Divide data into TRAINING, VALIDATION and TEST (in MNIST this is 50,000, 10,000 and 10,000 examples) VALIDATION is not needed 2. Encode the task into the algebra • Explanation to the the algebra of what we understand about the task • No hyperparameters	1. Divide data into TRAINING, VALIDATION and TEST (in MNIST this is 50,000, 10,000 and 10,000 examples) 2. Normalize data 3. Decide on architecture 4. Decide on activation function 5. Decide on optimization method and its hyperparameters 6. Decide on loss function 7. Decide on initial values of network parameters 8. Decide on method to avoid overfitting (dropout, regularization, early stopping using VALIDATION, and their hyperparameters) 9. Decide on batch size
LEARNING PROCESS	4. Run sparse-crossing several epochs. As there is no risk of overfitting, it can be run as long as you want and VALIDATION not needed. OUTPUT MODEL: created a set of e.g. 100,000 atoms	10. Run optimization algorithm several epochs (several times all data) until chosen termination method decides (see 8). 11. If result is bad in VALIDATION, go back to 1-8 OUTPUT MODEL: network with parameters found in this process
TESTING	MODEL used to assign test examples	MODEL used to assign test examples

Figure 4: Comparison between the use of AML and NNs. The key difference is that the many decisions required in NNs are not present in AML. Consequently, AML does not require a validation dataset.

We compared AML (without inductive biases) to NNs also devoid of inductive biases, specifically using multilayer perceptrons (MLPs). This comparison was performed on standard datasets: MNIST, FashionMNIST, CIFAR-10, and several medical imaging datasets. We evaluated families of MLPs with two, three, and four hidden layers. More precisely, we conducted a grid search over the number of neurons in the first hidden layer (512, 2048, or 4096 units) and the second hidden layer (256, 1024, or 2048 units), using Ray Tune [liaw2018]. When present, the third layer contained 128, 256, or 512 units, and the fourth layer (if present) contained 128 or 256 units. For the third and fourth layers, sizes were selected randomly. We ran 90 experiments using two-layer networks, 180 with three layers, and 90 with four layers, totaling 360 training runs. Each network was trained for 240 iterations or until the validation loss failed to improve for 10 iterations. In each run, we uniformly sampled the learning rate ($5 \cdot 10^{-4}$, 10^{-4} , $5 \cdot 10^{-5}$ or 10^{-5}) and the L2-regularization coefficient (10^{-3} , $5 \cdot 10^{-4}$, 10^{-4} or 0).

Results are presented in **Figure 5**. For each dataset, we show the distribution of test accuracies for the 360 MLPs, the test accuracy of the best-performing MLP on the validation set (blue vertical

line), AML result (orange vertical line), and AML followed by logistic regression using only the training data (red vertical line). In 2 out of the 10 datasets, AML underperformed relative to the best MLP but still fell within the performance range of the MLP ensemble. In 2 additional cases, AML combined with logistic regression achieved performance very similar to the best MLP. In the remaining 6 cases, AML outperformed not only the best MLP but also the entire set of 360 MLPs.

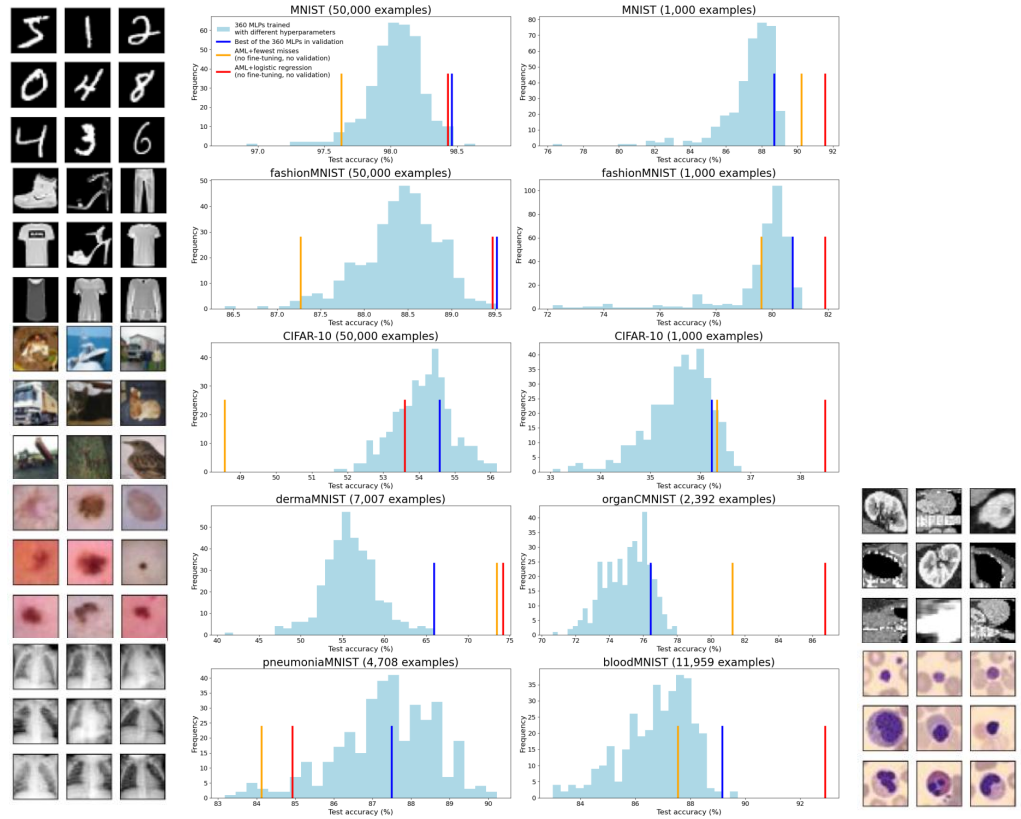


Figure 5: **Test accuracy of AML and MLP models on various image datasets.** We plot the distribution of test accuracies of the 360 MLPs, the test accuracy of the best MLP in validation (blue vertical line), AML (orange vertical bar) and AML followed by logistic regression (red vertical bar).

Learning from Formal Knowledge

So far, we have seen that the algebraic embedding approach and the subdirect decomposition of its models into atoms can be used to learn from examples. However, this method also extends beyond data-driven learning to cases where we work with axiom sets that describe a problem without containing any explicit examples.

We illustrate how AML can handle formal problems using the case of Hamiltonian cycles. The goal is to specify, through axioms, that we seek a closed-loop path visiting each node of a graph exactly once. Assume we have a graph with v nodes and e edges. To search for Hamiltonian cycles, we embed

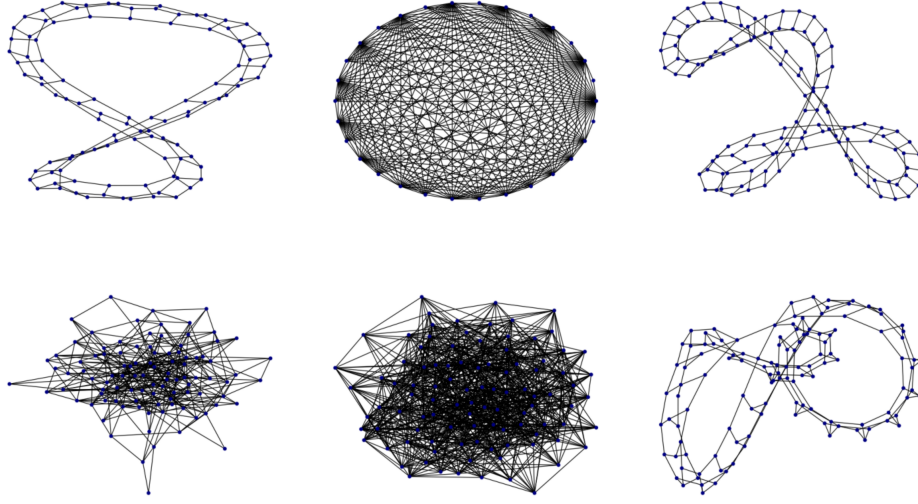


Figure 6: Hamiltonian cycles obtained using AML for different graphs. Sparse Crossing finds Hamiltonian cycles in randomly generated graphs with varying edge densities (first two graphs of the bottom row), modified Flower Snarks (SNm_124, bottom row, right), graph G7 of the FHCP set [FHCP1001] (top row, left), Sheehan graphs of various sizes (top row, center), and generalized Petersen graphs (GPN_122, top row, right).

the problem using the following constants: a constant V_i for each node, a constant E_k for each edge, a constant P to denote the path we aim to compute, and a constant W to encode constraints and optionally incorporate training results obtained during the cycle computation process. Additionally, we use: nE_k to denote the absence of edge k ; auxiliary constants Z_k for each edge; and context constants g_k and h_k for each edge. To ensure the path is connected, we include id_i constants, one for each node. Altogether, this yields a total of $2v + 5e + 2$ constants, where v is the number of nodes and e is the number of edges.

For example, we use the following axiom. For each node i and for each pair of edges E_y and E_z , we write:

$$P \odot (\odot_{x; x \notin \{y, z\}} nE_x) = E_y \odot E_z \odot P,$$

where the idempotent summation $\odot_{x; x \notin \{y, z\}} nE_x$ runs over the indices of all edges incident to node i , excluding edges y and z . This axiom specifies that, in the context of path P , the presence of exactly two edges (E_y and E_z) at a node is equivalent to the absence of all other edges at that node. This encodes the requirement that no more than two edges in P can be incident to the same node.

Once a model M of the embedding axioms is produced, we interpret that path P includes edge E_k if and only if $(E_k \leq P)$ is valid in the model.

AML successfully identified Hamiltonian cycles across a wide range of random and challenging graph instances (see **Figure 6**).

In summary, AML provides an alternative approach to Machine Learning using the subdirect product framework from abstract algebra. We have demonstrated how certain subsets of the subdirect product yield models with strong generalization properties and developed an algorithm to obtain them. Applied to standard datasets, AML produces results comparable to MLPs, and in some smaller datasets, it outperforms all 360 MLPs tested. We applied AML both to cases with examples (classification problems) and to a case involving only formal specifications without examples (Hamiltonian cycles). Each problem, of course, has its own set of axioms, but the learning process remains the same across all these cases.

References

- [1] Garrett Birkhoff. “Subdirect products in Universal Algebra”. In: *Bull. Amer. Math. Soc.* 50 (1944), pp. 764–768.