Computer Science and Information Systems 23(1):561–583 https://doi.org/10.2298/CSIS250425014G

Cascade Systems as An Implementation of a Gray-Box Architecture: A Case Study in Traceability for Microservice Scaling ⋆

Jorge Jiménez García[1], Ignacio Lacalle Úbeda[1], Paweł Szmeja[2], Katarzyna Wasielewska-Michniewska[2], Przemysław Hołda[2], Maria Ganzha[2], Carlos E. Palau Salvador[1], Costin Bădică[3], Stefka Fidanova[4], and Marcin Paprzycki[2]

[1] Universitat Politècnica de València, Valencia, Spain
jjimgar1@upv.es
iglaub@upv.es
[2] Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland
pawel.szmeja@ibspan.waw.pl
marcin.paprzycki@ibspan.waw.pl
[3] Department of Computers and Information Technology, University of Craiova, Craiova, Romania
badica_costin@software.ucv.ro
[4] Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Sofia, Bulgaria
stefka@parallel.bas.bg

Abstract. There is great potential in leveraging Artificial Intelligence (AI) systems to optimize complex infrastructures, automate difficult tasks, or support autonomy and coordination between networked devices. However, advances in state-of-the-art AI often neglect features and/or requirements that businesses care deeply about, namely traceability and explainability. A majority of available research has not explored much the deployment of semi-physical architectures combining fuzzy rule-based systems with more opaque models to improve explainability, being this specially true for the management of microservices in cloud and cloud-edge environments. This contribution builds on previous work that proposes a middle ground of mixed AI architectures that combine the performance of black-box AI models with "a more explainable overall architecture" by implementing a microservice scaling system for distributed cloud environments using a cascade approach. This work demonstrates and evaluates an application case of such an approach departing from a Service Level Agreement compliance, in a case of microservice scaling decision over cloud (and cloud-like) infrastructures.

Keywords: explainability, fuzzy-rule-based-systems, semi-physical architectures, AI network, gray-box

1. Introduction

Artificial Intelligence (AI) systems have an increasing impact across modern society, from personal life, where these systems can know songs one would like to hear, answer questions, or enable creativity by generating images; to the enterprise world, where AI is used

⋆ This paper is a continuation of a Conference Paper first published in INISTA 2024 as DOI 10.1109/inista62901.2024.10683838

to try to predict stock market trends, identify faults in assembly line products, or aid development of new materials, among many other uses.

Enterprise use cases, however, present additional requirements. While an average user might just be annoyed when recommended a book they did not like, the operator of the recommendation system may want to know exactly "what went wrong" in making the recommendation. This is doubly true for various infrastructures, where the impact of an AI error may be more significant than simple annoyance. If, for example, in the Smart City, the energy generation and provisioning is driven by an AI system and, as a result of a mistake made by that system, the city experiences a blackout, the citizens, officials, and grid's operator(s) have to know what happened and fix the problem if (and as soon as) possible.

Another relevant example is found in microservices allocation across a computing infrastructure. Cloud providers, and other related businesses, quantify their success on their capacity to efficiently dedicate their resources to the dynamic demand. Often, intricate models are used to support such decisions, that may lead to widely different levels of productivity and Quality of Service (QoS) to the final user.

Knowing why AI-driven systems obtain specific results is complicated and research in Explainable AI (XAI), is concerned with finding answers. Here, some systems are relatively simple to understand. For instance, rule-based or decision tree-based systems are inherently explainable, due to their design and inner workings. However, modern architectures like Neural Networks or Reinforcement Learning based models are not easily explainable. This concerns, in particular, large and complex models and/or models trained on large (and very large) datasets. Unfortunately, these inexplicable (black-box) models show significantly increased performance when compared to their simpler, explainable (glass-box) relatives. Therefore, a lot of research effort is devoted to explain why a given model "behaves the way it does". The work in this paper departs from a previous effort by the authors to conceptualize an AI system building paradigm based on the use of discrete, independently trained AI subsystems, combined to create a system with a complex goal such as microservice allocation, by leveraging independent elements in a cascade manner [1].

In the current work, authors go beyond and analyse deeply the potential applicability of XAI methods in cloud-edge environments, as well as illustrating the XAI-as-architecture design with a practical example. In particular, the options of using different techniques over AI models to allocate resources in a distributed environment illustrate that there are various possible avenues. From our perspective, exploring gray-boxed design in the ML algorithms that recommend the commissioning of workloads should prevail over opaque, end-to-end systems. Here, by leveraging discrete components, the goal is to increase overall explainability thanks to their known purpose(s), and to use their interrelations to bridge the performance gap, when compared to a purely black-box model. Such approach helps also to identify flaws in AI systems, by being able to identify which (compartmentalized) element(s) of the overall system is(are) causing errors, as well as increasing the ease of fixing by allowing integrators to more easily replace "faulty" segments of the network.

2. Related work

2.1. Characteristics and types of XAI

Analysing existing literature on the benefits and necessity of XAI, authors of [2], [3], deliver a comprehensive study, focused on regulation compliance, user trust, and the potential to develop, or gain, deeper understanding of the cases where AI systems are used. They also go over the types of information that are relevant for the different user profiles, their roles, and expectations, as well as the insight gained from each sort of explanation of the model, and the issues it can help identify. From the system operators viewpoint, it is possible to identify the understanding of general model behaviour and of individual predictions, as the critical aspects of explainability for the infrastructure, known as global and local explainability, respectively [4].

In another work, authors of [5] studied current approaches to XAI in an industrial context. They proposed an end user, which is conceptually similar to an infrastructure operator, with similar requirements and concerns. It is evident from their compiled studies that the majority of recent advances tend to use post-hoc methods for explainability, which attempt to explain the behaviour of inherently non-explainable models after training, instead of attempting to improve the glass-box models to match the other systems.

Besides this, [6] builds upon the computational analysis framework introduced in [7]. In it, the author differentiates levels of explainability in three broad categories - computational ('what is it doing'), algorithmic ('how is it doing it'), and implementational ('where is it doing it'). To further the reliability of AI models for industry, all of these categories must be accounted for. From the perspective of industry stakeholders, *what* is already accounted for, and *where* is relatively controllable with the advent of distributed compute. It is the remaining question then, *how*, that is the main roadblock for explainability - given the popularity of black-box predictors.

2.2. Explainability for black-box models

Methods for explainability in black-box models, in particular in the case of system operators, which are of core concern here, are: for local explainability covering predictions, Shapley values-based techniques (SHAP) [8], and for global explainability regarding the behaviour of the model, LIME-based methods [9]. However, studies of both methods come to the conclusion that they are not robust enough to be relied on, to deal with the industrial-world explainability concerns [10], [11]. Regardless, the currently accepted practice is still to utilize decoupled XAI system, and relying on post-explainability, meaning the outputs are only explained *a posteriori*. [12]. More broad, comprehensive studies, covering black-box models report that the current techniques are insufficient or lacking in delivering understanding of local and global behaviour(s) [13]. Additionally, many scholars have criticised the use of black-box models combined with explainability techniques, since more explainable models would suffice [14], [15]. Nonetheless, reasons for such solutions being still used, regardless of their limitations, may be related to strict requirements for a specific model architecture that cannot be influenced, but at the same time the necessity to include explainability in the system in any feasibility form. Decoupled and a posteriori explainability is most intuitive and easy to include without influencing the black box type system internal structure. Moreover, such architectural approach can be

treated as an optional "add-on" that from the administrative point of view may be easier to manage.

2.3. Explainability in microservices management over heterogeneous Cloud-Edge environments

The cloud computing paradigm, although predominant for the last decade or so, is increasingly widening its scopes, considering other -more heterogeneous- type of devices to support modern services execution. The continuously changing technological landscape has witnessed recent advances related to more intelligent devices capable of combining their capacities to perform tasks with varying demands. The previous, together with programmable networks, development of cognitive cloud systems and advances in orchestration across different cloud environments has resulted in a whole new set of challenges. Additionally, the above exposed advent of explainability opens up a new research field aiming at providing a better interpretable AI-based decisions with regards to microservices management (e.g., allocation, failure diagnosis, dynamic scaling, etc.).

In this context, specially in the last very few years, some literature can be found dealing with this matter. In 2021, [16] already envisioned the application of XAI metrics and visualization techniques to improve transparency and interpretability of cloud resource allocation (which was, in turn, commissioned by the result of generative models and reinforcement learning models). However, those techniques were mentioned, in abstract, without specification or demonstration.

In [17], the usage of XAI techniques was proposed over a simulated cloud-only environment to improve robustness and transparency. Departing from modelled cloud resources, it generates synthetic data and suggests the applicability of generic XAI characteristics over Generative AI models (Variational Autoencoders -VAEs- and Generative Adversarial Networks -GANs) that predict the modification of those resources after workload allocation. However, again, auhors did not propose any particular methods or tools to be applied, stopping at generic/theoretical level. Remarkably, the work did not explore the implications of moving to edge or IoT elements neither. In [18], authors went deeper and proposed a high level framework that depicts how to possible embed such XAI concepts into a Deep Q-Network-based resource allocation over cloud-only resources, dealing with a state, action reward tuples following a reinforcement learning structure. Again, concreteness regarding XAI is very scarce and assumptions are only discussed over simulated scenarios.

Notwithstanding, at a certain point specific methods and tools were commenced to be applied over cloud-native environments. In this sense, several XAI techniques exist and can be classified into global (explaining the entire model) or local (explaining specific predictions). Other authors also add classification depending on its agnosticity to the model and on the stage of development of such model (ante, post) [19]. Very recently, a comprehensive survey [20] exposed that explainable methods are currently instilled in microservices management (mostly, malfunctioning and failure diagnosis). Although not strictly related with the scope of this paper, such work illustrates how certain techniques such as causal inference-based methods (e.g., DoWhy, Granger causality) or rule-based or graph-based dependency tracing (e.g., service dependency graphs) can improve system-level explainability and transparency in cloud environments dealing with microservices.

As discussed, microservices malfunctioning was a major concern in the first practical utilisation of XAI in cloud environments. Here, the goal was to showcase to the human why such failures occurred, or highlighting which would be the causes based on ML predictions. For instance, Authors in [21] proposed a novel system (ShapleyIQ) based on Shapley weighted values to analyse cloud-data-center-like system diagnosis (https://github.com/lonyle/ShapleyIQ). The approach followed was to decompose a complex system into smaller pieces, and adding splitting invariance over the traditional Shapley values method. A positive note is the utilization over 15,629 operations on 86 database services, deployed on 2,546 machines in a relevant Kubernetes enviroment.

Furthermore, resources scaling in cloud environments was also a main research target. Cloud vertical scaling involves pre-defining and allocating specific compute resources (CPU, RAM, etc.) to a server/cluster/deployment based on known workload requirements, ensuring it can handle expected performance demands. Among other works, in [22] authors proposed a resource scaler for Kubernetes environments and the application of XAI methods to elaborate on the cases where, after such scaling, a service would still experience some degradation. SHAP (SHapley Additive exPlanations) is used to deduce the most important features that yield such degradation that was, in turn, predicted by an XGBoost model. The final goal was to understand the root cause in order to enable more efficient use of the resources available in the infrastructure, which has certain commonalities to the field under study in this paper (service allocation). In a similar line, [23] aimed as well at building an explainable autoscaling framework for Kubernetes based on custom controller and requirements based on user surveys, thus proposing a relevant evaluation framework on how to assess the explainability of such system. Also in the cloud scaling context, a relevant experiment was performed in [24], where XAI is inserted when using LLMs for a trustworthy anomaly detection in dynamic-scaling, cloud-native based 6G networks. In particular, a use case is presented using XGBoost as AI, SHAP as XAI, and Llama2 as LLM. After the experimentation, which is available in a video demonstration [5], authors outline that the combination of usual XAI tools (e.g., SHAP), which derive numerical results, with LLMs (that provide actionable, human-readable information) might remarkably improve the explainability and transparency. However, the work does not consider microservice allocation, and does not explore the heterogeneity of cloud-edge-IoT environments.

Significantly, the interest in including XAI techniques in cloud environments has not stopped at research practitioners. The reference [25] outlines the already on-going applicability of XAI techniques in usual cloud CI/CD processes, naming renowned cloud providers and commercial tools that are currently working in this line TensorFlow Explain, H2O.ai, AWS SageMaker or embedded XAI in Azure Machine Learning or Google AI Platform.

However, literature is much scarcer whenever considering heterogeneity in computing resources (i.e., not cloud-only). The fact of encompassing edge servers and other less capable devices has only been covered in a few works. In [26], a group of researchers presented ORRIC, a linear complexity online algorithm that does indeed consider computation-limited resources (e.g., edge servers), devising a proper way of allocating AI training and inference loads on such devices. While not including standard explainability tools, the work actually tests over CIFAR-10-C dataset in a real edge com-

[5] https://www.youtube.com/watch?v=1CoDNVWJcqA

puting environment. ORRIC focuses on data and model drift for resource allocation upon MobileNetv2 and RestNet50 models, improving the performance of existing approaches. Explainability-wise, since the model does not rely on neural networks or other opaque mechanisms, the procedure followed can be traceable. However, it does not focus on model or process explanation per se. Contrary to other publications, an open source repository is provided, allowing for experiment replication [6].

Still, one of the most relevant references for the study in question in our paper, is [19] where authors outlined which XAI methods would be more appropriate in cloud-edge orchestration cases. Services allocation in a distributed environment build upon the application of certain AI algorithms over a set of gathered data (monitoring). Such data (e.g., network status, CPU, RAM of the nodes, services/workloads characteristics, etc.) can be expressed in various formats. Authors suggest that, for models that address allocation based on tabular data, SHAP, LIME (Local Interpretable Model-Agnostic Explanations) and counterfactual facts would perform well. In cases where timeseries data is prominently used, recommended XAI methods would be Integrated Gradients, Grad-CAM (Gradient-weighted Class Activation Mapping) and DeepLIFT (Deep Learning Important FeaTures). Finally, those cloud-edge orchestration models on top of raw text data (e.g., NLP or LLM) should be focused on explaining where the attention mechanisms have been placed.

As a conluding remark, while some insights on explainability and evaluation of explainability are available in cloud environments, they only cover a broad spectrum of applications, limiting its deployment in cloud-edge cases to a few isolated examples.

2.4. Gray-box models and the cascading approach

The in-between of explainable models (neither black- nor glass-box) are referred to as gray-box models. Typically they combine black-box models with different techniques that increase the explainability, without significantly decreasing the accuracy [27]. For instance, [28] starts with a Convolutional Neural Network (CNN) which can detect if a drain is obstructed or not. In an effort to improve the explainability, the authors propose to segment the behaviour of the network, devising a Rule-Based System for classification, once certain elements have already been detected within the images. The intermediate element symbolizes the "reasoning" of the system; i.e. if a drain is detected to be obstructed, it can be explained because there were a number of obstructing elements detected in the image.

Interestingly, in [29], the opposite strategy is adopted, where the authors develop a Guided Bayesian Optimizer (GBO) for the memory allocation. Here, a white box model is used first, to refine the incoming variables, that are then fed to a black box Bayesian Optimizer (BO). In terms of explainability, the refinement of the input variables leads to an increased understanding of the underlying dynamics of the memory allocation, and is reflected on the lower time-to-converge, accuracy of parameters, compared to an exhaustive configuration search and the inference time, compared to a RL algorithm. Authors also compare these systems to RElM, a fully white-box expert system they developed that performs significantly better, ultimately highlighting that incorporating expert knowledge into these algorithms improve their performance. In a similar manner, [30] separates

[6] https://github.com/caihuaiguang/ORRIC

the face detection process in a two step approach, to leverage Support Vector Machines (SVM), tasked with detecting the separate parts of a face, and a geometry model that takes in the detected parts and assesses if they form a face. This explicit separation benefits the model in significantly improving its robustness, as well as indirectly improving explainability.

Note that these approaches do not make the models completely interpretable, but rather make them "*more interpretable*" than a pure CNN or BO approaches, while retaining the performance benefit of these types of deep learning architectures.

Further, works such as [31] envision graphical-based explainability methods resistant to online updates, and crucially, mostly decoupled from the inference and framework, such that the additional cost for explainability is reduced as much as possible. Given this graphical structure, devising a graph structure for AI systems in a cascade-way seems to be helpful to further explainability. Obviously, this approach is only feasible if we have intermediate computations to base our cascade predictors on. More generally, architectural explainability builds on the previously mentioned computational analysis framework, accounting for the 'how' of explainability: the chain of operations to obtain the output.

2.5. Semi-Physical Network architectures

Exploring explainability in complex models, [32] introduces the idea of a Dynamic Semi-Physical Network. A Semi-Physical Network uses preexisting knowledge of a domain space to model the internal connections of a Neural Network system, composed of distinct subnetworks to reflect known relationships, found in the real world. In general, it can be stated that, for a function that is the result of several intermediate functions that have some sort of relation, the network that represents the interconnection of these (intermediate) functions would be more understandable than a fully black-box model, by making certain connections explicitly known. An example of such a representation can be seen in Figure 1. The authors also assert that more detailed (and faithful) model structures, matching real world relationships, result in increased performance from the system, delivering benefits of applying expert knowledge. Note that the intermediate functions do not necessarily have to be Neural Networks. They can be directly modelled by mathematical functions, or other estimators.

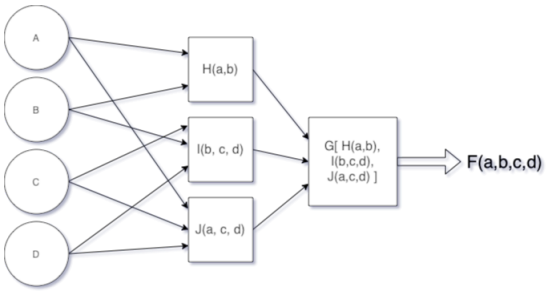

Fig. 1. Semi-Physical representation of function $F(a, b, c, d)$, that is composed of $H(a, c)$, $I(b, c, d)$, and $J(a, c, d)$

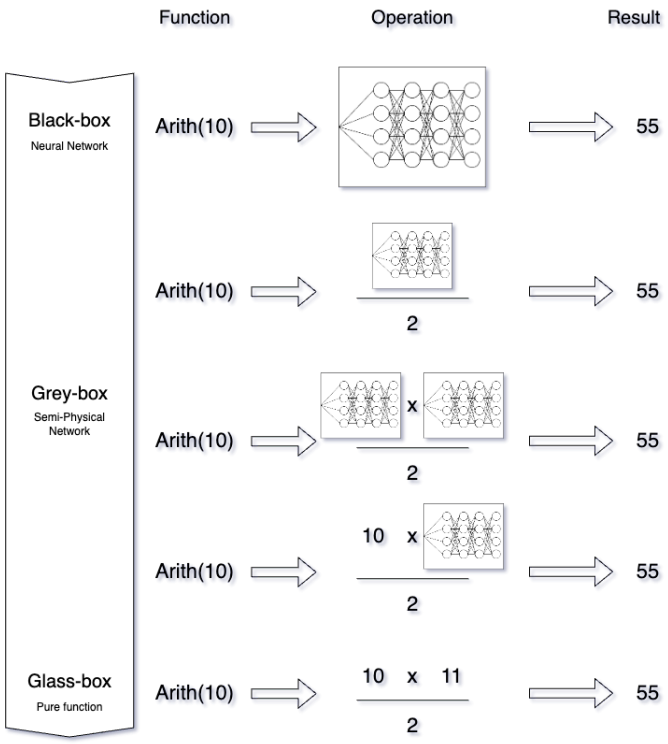

Fig. 2. Gradient of explainability showing different model architectures to calculate the arithmetic sum $1 + 2 + 3 + \cdots + 10$

This approach is supported by research reported in [33] that analyzes the sparsity of parameters in Neural Networks, which can be understood to represent the intermediate functions mentioned previously. Additionally, researchers show that, since this relationship is structural, improvements to NNs through pruning cannot be learned from scratch. In a number of studies, Semi-Physical Networks are successfully applied and shown to outperform NN approaches [34]–[36]. In this way, Semi-Physical Networks provide a middle ground between black and glass-box architectures (See Figure 2)

2.6. Fuzzy Logic for Semi-Physical Networks

Fuzzy logic [37] is defined as a logic system where a truth value can take any real number between 0 and 1, as opposed to the Boolean logic, which only allows the discrete values 0 and 1. Applied to Semi-Physical Networks, this can be used to help describe the in-between intermediate functions that are nebulous or uncertain. Conceptually, it can be seen as knowing what level of relationship or dependence exists between an intermediate function and the final result, without knowing what the relation actually is. Fuzzy rule-based systems (FRBS) are a kind of rule-based systems that aims to capture the uncertainty of the represented knowledge [38] by assigning a degree of truth to the

rule predicates, which may also be overlapping instead of the binary rule-based-systems. An example is shown in Figure 3. This flexibility allows a FRBS to address non-linear problems, such as in Neural Networks, with the inherent explainability, transparency, and determinism of the rule-based-systems.

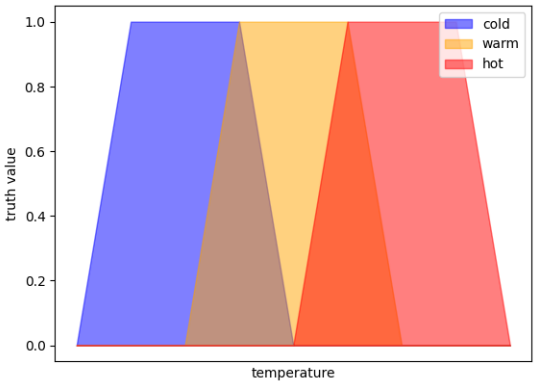

Fig. 3. Example of fuzzy set for 'temperature'. The discrete terms have overlap among them, representing their uncertain threshold, which are auto-determined.

This means that the obtained rules are as globally explainable as they can be, due to their deterministic nature and explicit representation of knowledge [39], [40], while still having a satisfiable degree of local explainability thanks to the fuzzy determination of predicates. In [41], a Fuzzy Semi-Physical Network is successfully developed, for approximating properties of Lithium-Ion based batteries. Fuzzy Semi-Physical models consistently outperform their non-fuzzy and non-semi-physical counterparts, by using a FRBS regressors as approximators of the unknown intermediate function in a model [42], [43]. Going back to the example in Figure 1, one can imagine that H and J could be approximated by a fuzzy estimator instead of by a mathematical relation.

3. Architecture of a Cascade Gray Box System for Replication in Cloud Infrastructure

A specific use case for Dynamic Semi-Physical Network can be inspired from the container allocation research explored in 2.3. In particular, concerning the problem of cloud container scaling. In the context of cloud applications, developers deploy and make available their programs, or services, on provider's servers, so that they don't have to deal with the cost, or the complexity of acquiring and managing their own server infrastructure. Often, service deployers own several clouds (or cloud-native intrinsic [44]) resources that can be commissioned to run such containers. An additional aspect influencing the cost and features for operator is the number of replicas, or mirrors of the deployed service, are available at any given time. Current solutions rely on reactivity, modifying the number of

replicas when usage hits predefined thresholds, rather than using a predictive approach [45].

Dynamic selection of the proper level of replication is paramount in achieving the promised performance and latency for the users. Here, rule-based systems are widely utilized, by filtering the available capacities, e.g., bandwidth, CPU, pod scalability, and reacting based on the current state of the service deployment. However, infrastructure owners may want to increase the productivity of their equipment (physical or virtual) by turning to AI systems to manage the replication level of their client's application. Keeping in mind that Cloud Providers typically have some Service-Level Agreement (SLA) that dictates the conditions of reliability that the provided resources have to the client, this poses a key question for the proposed AI system and the service owners:

"If the SLA is broken, how do we audit our system to investigate what went wrong?"

Generally, the state-of-the-art systems rely on Reinforcement Learning (RL) Neural Networks to handle allocation, with variations on the specific kind of RL system [46], [47]. RL models are a good fit for systems in which one wants to perform sequential decision-making, where actions affect not only immediate outcomes but also future states, as is the case when creating additional replicas over an entire cloud infrastructure. However, using these kinds of systems (that still rely on black-box models) presents crucial difficulties in answering the auditing question, when they fail.

Despite the difficulties, it is still possible to achieve some level of explainability in black-box based systems. A possible strategy is to enable a "plug-in" explainability step following the Function-as-a-Service (FaaS) approach – with an external service called the explainer. Here, for example, one may use SHAP values to provide explainations to the decisions being taken. In order to prepare such a solution, one needs to decide which part of the system (for instance, a model that allocates a set of interconnected tasks to a set of computing nodes) should be explained, e.g., a policy network. Then, the explainer assumes that the selected part behaves as a function in which each input feature's contribution to the output will be assessed. Additionally, access to a representative (ground) dataset is needed to prepare the explainer. Finally, examples to be explained can be sent by the AI-based service to the explainer service. However, the produced explanation in its raw form is purely numeric, which can be time-consuming for a human to inspect. Therefore, other ways of presenting the final SHAP output can be employed, such as an importance graph that gives insight into the relative significance of input features influencing the output, i.e. insight into which attributes where most significant when making the decision.

The described approach can be applied to a wide range of different tasks, and the instances of the explainer are easily scalable. However, apart from the previously mentioned limitations of the SHAP values, the solution may not scale well with respect to the number of the input features explained. Moreover, the behavior of the explainer is sensitive to the choice of the ground dataset. Finally, the solution can be resource-intensive, limiting its use in edge environments. From the aforementioned considerations it can be seen that before choosing such an approach some preliminary analysis focused on practical considerations should be performed.

3.1. Allocation and Replication in Cloud-Edge Infrastructure

Cloud Allocation and Replication is a very complex topic, which we will cover briefly. Although cloud servers usually share common traits, i.e., interfaces homogeneity, high availability, or built-in protection, addressing their dynamic status is still a hot research topic [48]. Escalating in difficulty, whenever other computing elements come into play, additional aspects such as network reliability, resources availability and format heterogeneity must be considered [49]. This happens especially when a complex, distributed system of computing nodes is put together in a so-called "computing continuum". There, a myriad of devices ranging from smart IoT assets to fog nodes, or edge computing equipment, are included [50], [51]. Furthermore, when restricting to the case of replication (which can be considered as an allocation of a service identical to an already-deployed service), some information about the state of the original service is available to help guide the creation of the new replica, namely the usage of the original service. In this context, we also have additional needs that influence the replication process, such as a system that determines when the replication is needed. To improve over current solutions in terms of performance, predictive elements can be of use here. Additionally, explainability is always a goal keeping in mind for situations where infrastructure failure is a possibility.

Given the compute limitations of nodes in an IoT or fog continuum, reaching a point where we reactively need to replicate a service is not desirable, since this restriction in compute means the latency to rescale imposes an unacceptable delay on the user's service.

First, some logical assumptions need to be made. In general, if one has several machines in the server side, one would not want to have a particular, busy machine have another allocation on it when there are unused machines available. Second, and specific to service replication, is resource availability. Given that we expect (or even have run against) resource limitations with the current set of replicas, we require plenty of available resources for our soon-to-be-created replicas. This resource limitation question is where predictive elements may be of use, establishing a bound on the resources the replica needs, ensuring efficient use of the limited resources on the network-wide continuum. Additionally, this predicted bound may be lower than the current level of usage, indicating a prediction to scale down the service.

There may be other metrics of relevance, like the network traffic of the machine, but this should prove sufficient for a sample example of a simple network.

$$\Delta Replica(...) = Best[C \mid C_{Future}(...), R \mid R_{Future}(...), ...] \tag{1}$$

In this case, the CPU and RAM usage are available for the variables that have been established to be relevant to make the decision. However there is no way of knowing their future value, thus, they are the intermediate functions for a decision, an the estimate of the future state of the machines. To predict these future values, one can also assume that an intermediate function may use previous measurements to estimate future usage. These values can be supplied to a final function that estimates the optimal number of replicas (or the change in number of replicas, Replica Δ), which is then enforced through K8S' Horizontal Pod Autoscaler (HPA). The sample proposal is shown in Figure 4, and is described in short in Eq. 1 where C_{Future}, R_{Future} are predictive intermediate functions for the future values of CPU and RAM usage, respectively, $Best$ is a selection algorithm that optimizes given arguments according to some criteria, and "Trigger Replication?" is a rule-based system.

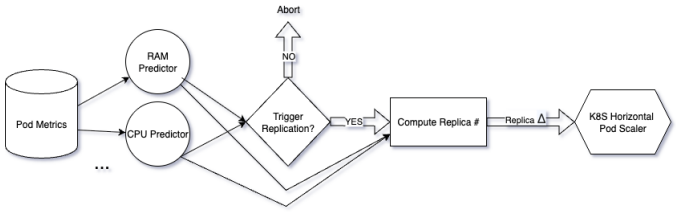

Fig. 4. System Diagram for the proposed Gray-Box Horizontal Pod Autoscaler

It should be stressed that the implementation of these intermediate functions could be a black-box model. Furthermore, systems such as the rule based system that determines if the replication should trigger could be automatically determined through Fuzzy-Rule Based System (FRBS) or even set manually by the user, including a human-in-the-loop element in the system and enhancing explainability.

3.2. Characteristics of an intermediate estimator

In the cases where the intermediate function is an estimator or model, there are some important characteristics that they must have, to maintain the desired properties in the system, such as being relatively explainable and self-contained.

First, note the benefit of having some degree of explainability on them, by knowing *a priori* the role the component plays in the overall system. The fact that the explainability is derived from knowing the intermediate function's role also hints at an interesting constraint: these intermediate functions have to be pre-trained to be included in the system. This has some benefits, like being able to individually evaluate their performance or bias, without compound error, or being able to choose different implementations, based on a performance-to-explainability or performance-to-relevance tradeoff. In our example, we train individual CPU and RAM predictors, which are locally explainable given that we know their function in the system, and that do not have compound error at this stage in the system because they are the first stage of the replica scaling process. Additionally, they can be trained independently of the rest of the system, and thus upgraded and replaced.

However, the same way that the compound or cascade error is not a factor for the individual components close to the start of the system, intermediate components that depends on others, i.e., RL systems, may imply a certain error propagation, or an accentuated dependence on the behaviour of such subcomponent. It is important to bear this in mind. in order to favour modularity of the system, and to minimize the codependency of subcomponents, as well as influencing, which kinds of subcomponent systems may be a good fit. Ideally, intermediate components should favour glass box models to minimize cascade error as much as possible. In our example, "Compute Replica #" is a mathematical, glass box system. Continuing with the themes of the initial predictors, this component is also locally explainable, with a defined function. Finally, since it depends on the previous predictors and is a mathematical (and thus not trained) model, upgrading previous parts of the system in which we depend on improves this component's accuracy, since there is no codependence between them.

3.3. The lifecycle of a component-based Semi-Physical System

The component-based perspective brings also advantages, in terms of maintaining or upgrading the system, which is a crucial factor to retain competitive advantage for the operators. Given that the Semi-Physical Network is a software-defined-architecture (SDA), it can be modified at any time, to reflect the changes in the knowledge dynamics, which are reflected in the internal structure of the system, as well as enabling very easy addition of new information, or intermediate functions, to the system. Therefore, it can easily be refined further, which is a key factor in providing speed and agility, when compared to other full black-box or monolithic systems, which would have to be rebuilt, or retrained, from scratch.

This is not only applicable to the complete system, but also to the intermediate functions, similar to a microservice architecture, a structure the software industry is quite used to. Intermediate functions can also be easily upgraded, replaced, and remade, provided the previously established constraints, especially self-containment. This comparison also extends to the development of the estimators, where several different teams could develop and own the components, as it is commonly done for their microservice counterpart. Additionally, although it may be detrimental to the explainability, it enables operators to use complete black-box models as their intermediate estimators, if they wish to, or require, the increased performance.

3.4. Traceability and Explainability in Gray Box Cascade systems vs Black-Box systems

Traceability in Dynamic Semi-Physical systems lies in the causal relations between the intermediate functions and the final output of the system. Continuing with the example, if a given service fails to uphold the SLA requirements, for example a responsiveness requirement, the initial trace may look first at the the estimated replica count (which is a glass-box model), from there, one can easily link to which individual predictor(s) (CPU, RAM, or both) failed, which is also easily debuggable because we have access to the real vs predicted values for that particular metric. Additionally, notice that this approach is independent of the particular systems used in our whole cascade: if the replica count calculation was model-driven and black-boxed, classical explainability techniques could apply to estimate the importance of features and backtrack to previous system elements.

In global terms, since the Semi-Physical system's architecture is expert-defined, it should be decidedly more understandable, as opposed to a dense NN-style architectures, where the underlying sparse structure is determined over several iterations of the nebulous gradient updates.

Furthermore, traceability becomes mire in a black-box NN system, due to the monolithic nature of that system. While explainability techniques like SHAP – which attempts to explain the effect of each input value on the final result – may be used, the phenomenon of NN coadaptation hinders traceability, due to the unforeseen internal dependencies in the network among inputs [52]. It is important to note that while the coadaptation may also be present in Semi-Physical Networks, it is somewhat self-contained in the different intermediate function estimators, rather than materialize throughout the entire system.

4. Self-Scaling: a Gray Box Cascade System for Replication Management in Kubernetes

4.1. Experimental Setup

To validate the development claims of Gray Box systems, we develop Self-Scaling, a custom cloud-native software influencing Kubernetes' features and parameters, to implement the design proposed in Fig. 4. As above mentioned, the goal is to intelligently, autonomously manage the replication of services (i.e., number of pod replicas of a service in a Kubernetes cluster) under the works performed in the Horizon Europe project aerOS [49]. For this system, we implement the notion of the Gray Box system by utilizing a cascade approach in model design, where independent components are interconnected in a left-to-right manner. System information from the pods managed by our custome software is used as primary input, in this case only CPU and RAM as starting metrics to track, as outlined in the original proposed system, though this could be expanded. Note that given we are in a Kubernetes cluster, some resources such as CPU are measured in Kubernetes' specific units, intended to harmonize the metric across heterogeneous compute nodes [53]. In keeping with the idea of having discrete components for each intermediate estimator, future timeseries models for both metrics are implemented independently of each other, utilizing ARIMA/AR-NET for timeseries regression [54] [55] [56]. This implementation uses Neural Prophet [57]. The decision component ("Trigger Replication?" in Fig. 4) is set by the service deployer in order to replicate the existing Kubernetes Horizontal Pod Autoscaler (HPA) behaviour as closely as possible, but in a more automated and intelligent way (i.e., predicting the required demand). As outlined previously, and as example of the modularity of a Gray-Box system, this component could be replaced with a FRBS module with minimal impact to other downstream components. Finally, to reduce the number of opaque or black-box models as we advance downstream in the cascade, "Compute Replica #" is implemented as a purely mathematical model based on margins, where the replica number is optimized to maintain a set resource margin available for every node in which the service is replicated, 75% in this case. To recap, Self-Scaling utilizes a predictive black-box component to estimate future usage of pods in a Kubernetes cluster, which is then handed to a rules module that can be user-set or automatically defined (and thus is a white-box in the case of being user-set, or gray-boxed in the case of an automatically defined FRBS), and finally utilizes a purely-numerical, white-box system to compute the target number of replicas. This system then communicates the new number of replicas to the Kubernetes Control Plane.

The test case chosen to evaluate the performance of Self-scaling consists of an API-based system implementing a single-task job submission service, managed by the proposed solution. This deployment is repeated afterwards and managed only with the default Kubernetes HPA. In order to simulate user access to the job queue by this service, several user profiles have been defined to simulate the anticipated behaviour interactions. User profiles are implemented as `cronjobs`, which submit computing jobs of random complexity via `curl` on a set schedule depending on the user behaviour they replicate. The number of users simulated and their schedule are defined in Table 4.1, and are described as follows:

- `burst-morning`: A high influx of jobs at the beginning of the workday.

Table 1. User simulation counts and schedules

Profile name	User count	Cron schedule
`burst-morning`	100	`0 9 * * *`
`whole-day`	50	`*/1 9-17 * * *`
`burst-evening`	80	`0 15 * * *`
`evening`	20	`*/1 15-20 * * *`

- `whole-day`: A regular influx of jobs throughout the day.
- `burst-evening`: A moderate influx of jobs at the beginning of the evening work hours.
- `evening`: An additional, constant influx of jobs during the evening work hours.

Data for both scaling managers are collected during a 24 hour period. Considering the overlap in user profiles, the modelled user influx is displayed in Fig. 5. According to its distribution, the default HPA is expected to decrease its performance or even hit the maximum number of allowed replicas with the sudden influx of users at 09:00 or 15:00, because it cannot anticipate to them. Additionally, since the jobs have varying degrees of complexity within a range, it is expected that the jobs will start to accumulate in a queue, given that the system will not be able to keep up with them, resulting in usage numbers different than the expected influx, since they cannot be processed in time before another round of jobs is submitted.

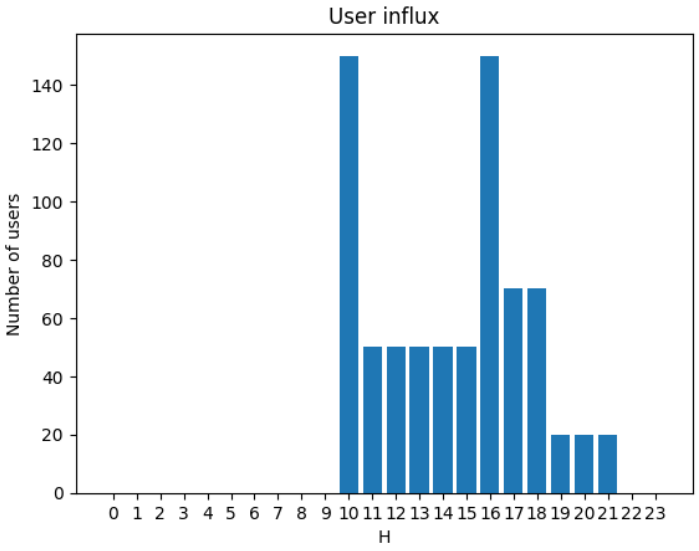

Fig. 5. Modelled user job influx

4.2. System Evaluation

Results from the tests performed in the experimental setup are presented and compared with the predictions returned by Self-scaling.

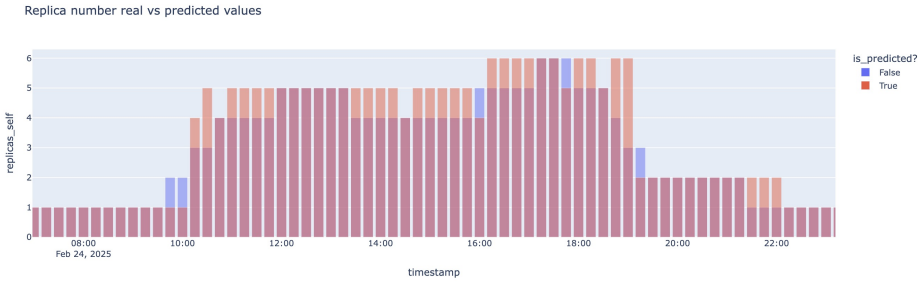

Fig. 6. Replica number in the HPA experiment (blue) vs Self-scaling predictive values (red)

In Fig. 6, the number of replicas obtained in the experiment is compared to the number predicted by Self-scaling. It is important to note the frequency of the data, which follows a default step of 15 minutes. For the default implementation of the HPA, scaling is evaluated every minute, and therefore it is not possible to see the sharp number of replicas that the HPA instantiates until reaching the required number. Regardless, this comparison demonstrates the predictive nature of Self-scaling, which for example anticipates a long-term rise in the number of required replicas at 11:00, before that increase is required later at 12:00, or notice how the predictive element accounts for the expected increase in users at 17:15.

This behaviour highlights the trade-off between resource efficiency and system readiness and reactivity, with our Self-scaling solution increasing the number of replicas way before they are actually needed. This implies that the system incurs in additional costs when the predicted usage rises above the actual one, since Self-scaling is forcing Kubernetes to spin up more replicas, but benefits the end users in terms of reactivity and system responsiveness.

4.3. An Example of Traceability and Debugging in Gray-Box Cascades

To showcase how to trace errors with Gray-Box Cascades, we return to Figure 6, in particular to the increased number of replicas in the interval between 18:45 and 19:00, or drops in the replica count in the period 14:30 to 17:45, where the standard Kubernetes HPA maintains a high number of replicas but Self-scaling has sudden reductions.

To trace these errors, given we know how the decision process is implemented, we work backwards from the scaling order to the ”Compute Replica #” module, which recall is a clear box mathematical model. Given it is clear box, we can see that it is implemented as a simple calculation that aims to maintain a maximum of 75% usage in all replicated pods, and emits a reduction if not. Since we do not know at this time which pod variable caused the replication, we work backwards to ”Trigger Replication?”, a rule-based

system, in this case user-set. The define ruleset is, in tune with the replica computation, defined to trigger if any of the predicted future variables exceeds 75%. Given this, we can now establish and diagnose the reason for each of the replication up/downscaling events. In this case, we focus on the period previously outlined, shown in Figure 7.

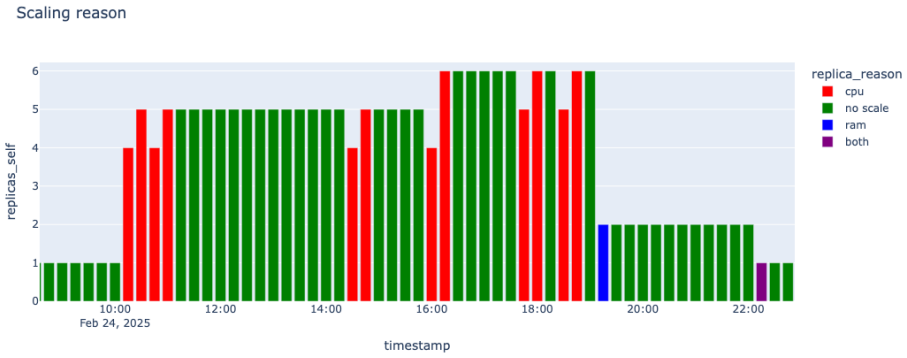

Fig. 7. Scaling reason for the Self-scaling manager

Noticing that all undesirable downscaling events were triggered because of the CPU rule, we can now look at the CPU predictor to further quantify the problem, which at this stage could still lie in any of the modules involved with CPU usage in the decision: the CPU predictor and its associated, user-set rule.

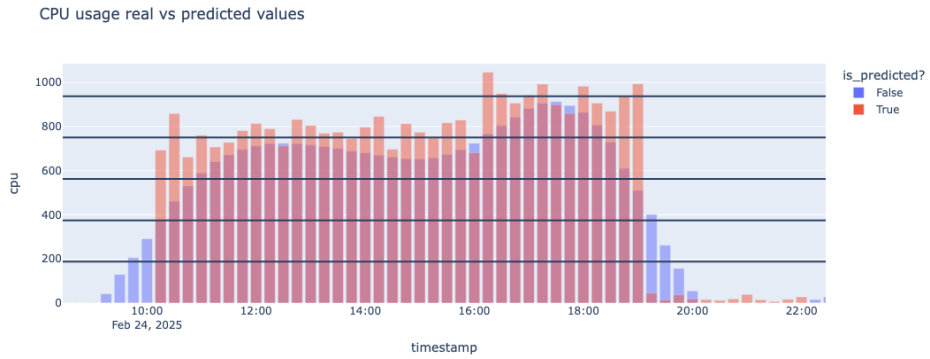

Fig. 8. CPU Module real values (blue) vs Self-scaling intermediate predictor (red). Horizontal lines marked at the up/downscale thresholds

Further inspecting and comparing the predicted CPU usage values and the rule thresholds in Figure 8, we can clearly identify points in the graph where two issues occur: the

predicted value falls under the threshold point, triggering a downscale, and it is also underneath the real observed value in blue. Given this discovery, and considering that the rule threshold worked well for all other instances, we can diagnose the issue to be a lack of accuracy in the CPU predictor module. Recall that this module is implemented as an AR-Net predictor, which is a mathematical model that uses Neural Networks and Gradient Descent to estimate the coefficients of the equation. While we could go further back in the Cascade, given that AR-Nets are somewhat interpretable, with this module being at the very start of the cascade we find no need. A system integrator may now take action having identified the root of the issue, for example expanding the data used to train the model, or deciding for a more complex black-box now that the need for performance is justified.

5. Limitations of the considered approach

A first obvious roadblock is the need to determine an overall architecture for the Semi-Physical Network. Since it would be informed by the knowledge of the field, a lack of such knowledge means an inaccurate estimate of the relationships of the intermediate functions, which may result in results worse than using a completely black-boxed model. Providing incorrect, or unnecessary, intermediate functions may also be detrimental to the overall system, as it would be fed misleading variables. As a result explainability is tightly coupled with the architecture which may be considered as harder to maintain in systems that may require any adaptations with time. On the other hand, the alternative approach with an external explainability service is easier to maintain as more detached from the model itself, however more difficult to return a meaningful result.

Furthermore, due to the inherent move to a gray-box approach, the approach is also constrained for the industry applications with strict explainability constraints, stemming from the legal or the operational requirements. As previously outlined, the proposed style of architecture only obtains *more explainable* models, as opposed to completely explainable ones. Here, however, it should be noted that requirements related to the degree of explainability should be collected at the beginning and considered during the design of the architecture. The gray-box approach may prove still valid in numerous cases.

Finally, traceability is not guaranteed, depending on the different interconnections between the subcomponents. While it is a reasonable expectation to use more explainable "aggregators" (the components that combine the inputs of intermediate functions) the architecture by itself does not guarantee traceability when using one, due to the presence of coadaptation internal to the aggregator component.

6. Concluding remarks

Drawing from the AI explainability theory, achieving transparency on where a model failed or behaved in a suspicious way that needs to be investigated, implies several degrees of difficulty. Current approaches are mostly limited to SHAP weights allocation and applying gray-box modelling to a certain extent. With the advent of Dynamic Semi-Physical Architecture (which authors leverage from different sources), one step beyond is envisioned.

This work proposes a cascading approach, that bids for gradually increasing the "transparency" of an AI system to be more explainable for the human in charge of understanding potential faults in the result. It does so by suggesting the introduction of intermediate aggregators, rooting on previously known relationships between inner blocks of the architecture. That way, a Semi-Physical Architecture incrementally approaches gray-box fragments, which will, in turn, be more traceable and dynamically analysed, compressing and reducing the scope of black-box type models (such as NN or RL) in those systems.

A direct conclusion is that, while this appears as suggestive roadmap, it must be validated with real test cases. In this regard, authors have proposed a simple but quite illustrative case where cloud-native services require intelligent policies for replica creation (based on some ML process). Although the results are good, authors are already working to substantiate the claim via experimenting on the proposed example of service and resource allocation possibilities in an IoT-edge-cloud environment. The results of these experiments, performed within the scope of various research projects (such as aerOS and O-CEI), will be reported in subsequent publications. Moreover, the underlying software being produced will be released under Open Source licenses.

Acknowledgments. This research was funded by the European Commission, under the Horizon Europe project aerOS, grant number 101069732. Work a result, in part, of a bilateral collaborations between the Polish Academy of Sciences and the Bulgarian Academy of Sciences, and between the Polish Academy of Sciences and the Romanian Academy. Tests were supported by Wojciech Buchwald.

References

1. W. Samek and K.-R. Müller, "Towards explainable artificial intelligence," in Explainable AI: Interpreting, Explaining and Visualizing Deep Learning, W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Müller, Eds. Cham, Switzerland: Springer International Publishing, 2019, pp. 5–22. ISBN: 978-3-030-28954-6.
2. R. Dwivedi, D. Dave, H. Naik, et al., "Explainable ai (xai): Core ideas, techniques, and solutions," ACM Comput. Surv., vol. 55, no. 9, Jan. 2023, ISSN: 0360- 0300. DOI: 10 . 1145 / 3561048. [Online]. Available: https://doi.org/10.1145/3561048
3. A. Rai, "Explainable AI: From black box to glass box," Journal of the Academy of Marketing Science, vol. 48, pp. 137–141, 2020
4. I. Ahmed, G. Jeon, and F. Piccialli, "From artificial intelligence to explainable artificial intelligence in industry 4.0: A survey on what, how, and where," IEEE Transactions on Industrial Informatics, vol. 18, no. 8, pp. 5031–5042, 2022. doi: 10.1109/TII.2022.3146552.
5. K. E. Mokhtari, B. P. Higdon, and A. Başar, "Interpreting financial time series with SHAP values," in Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering, 2019, pp. 166–172.
6. M. T. Ribeiro, S. Singh, and C. Guestrin, "Model-agnostic interpretability of machine learning," arXiv preprint arXiv:1606.05386, 2016.
7. E. Galinkin, "Robustness and usefulness in AI explanation methods," arXiv preprint arXiv:2203.03729, 2022.
8. A. Gramegna and P. Giudici, "SHAP and LIME: An evaluation of discriminative power in credit risk," Frontiers in Artificial Intelligence, vol. 4, p. 752-558, 2021.
9. V. Hassija, V. Chamola, A. Mahapatra, et al., "Interpreting black-box models: A review on explainable artificial intelligence," Cognitive Computation, vol. 16, no. 1, pp. 45–74, 2024.

10. C. Rudin and J. Radin, "Why are we using black box models in AI when we don't need to? A lesson from an explainable AI competition," Harvard Data Science Review, vol. 1, no. 2, pp. 10–1162, 2019.
11. C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," Nature Machine Intelligence, vol. 1, no. 5, pp. 206–215, 2019.
12. K. Arendt, M. Jradi, H. R. Shaker, and C. Veje, "Comparative analysis of white-, gray- and black-box models for thermal simulation of indoor environment: Teaching building case study," in Building Performance Analysis Conference and SimBuild: Co-organized by ASHRAE and IBPSA-USA, ASHRAE, 2018, pp. 173–180.
13. D. Thakker, B. K. Mishra, A. Abdullatif, S. Mazumdar, and S. Simpson, "Explainable artificial intelligence for developing smart cities solutions," Smart Cities, vol. 3, no. 4, pp. 1353–1382, 2020, ISSN: 2624-6511. doi: 10.3390/smartcities3040065. [Online]. Available: https://www.mdpi.com/2624-6511/3/4/65.
14. M. Kunjir and S. Babu, "Black or white? How to develop an autotuner for memory-based analytics," in Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD/PODS '20, ACM, May 2020. doi: 10.1145/3318464.3380591. [Online]. Available: http://dx.doi.org/10.1145/3318464.3380591..
15. Y. Oussar and G. Dreyfus, "How to be a gray box: Dynamic semi-physical modeling," Neural Networks, vol. 14, no. 9, pp. 1161–1172, 2001.
16. T. Gale, E. Elsen, and S. Hooker, "The state of sparsity in deep neural networks," arXiv preprint arXiv:1902.09574, 2019.
17. U. Forssell and P. Lindskog, "Combining semi-physical and neural network modeling: An example of its usefulness," IFAC Proceedings Volumes, vol. 30, no. 11, pp. 767–770, 1997. doi: 10.1016/S1474-6670(17)42938-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667017429387.
18. J. Adou, A. Brou, and B. Porterie, "Modeling wildland fire propagation using a semi-physical network model," Case Studies in Fire Safety, vol. 4, pp. 11–18, 2015.
19. L. Haviez, R. Toscano, M. El Youssef, S. Fouvry, G. Yantio, and G. Moreau, "Semi-physical neural network model for fretting wear estimation," Journal of Intelligent & Fuzzy Systems, vol. 28, no. 4, pp. 1745–1753, 2015.
20. L. A. Zadeh, "Fuzzy logic," Computer, vol. 21, no. 4, pp. 83–93, 1988.
21. L. Magdalena, "Fuzzy rule-based systems," in Springer Handbook of Computational Intelligence, 2015, pp. 203–218.
22. P. Mishra, "Model explainability for rule-based expert systems," in Practical Explainable AI Using Python: Artificial Intelligence Model Explanations Using Python-based Libraries, Extensions, and Frameworks, Springer, 2021, pp. 315–326.
23. J. M. Mendel and P. P. Bonissone, "Critical thinking about explainable AI (XAI) for rule-based fuzzy systems," IEEE Transactions on Fuzzy Systems, vol. 29, no. 12, pp. 3579–3593, 2021.
24. L. Sánchez, I. Couso, and M. González, "A design methodology for semi-physical fuzzy models applied to the dynamic characterization of LiFePO4 batteries," Applied Soft Computing, vol. 14, pp. 269–288, 2014. doi: 10.1016/j.asoc.2013.03.020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S156849461300135X.
25. I. Rodríguez-Fdez, M. Mucientes, and A. Bugarín, "Fruler: Fuzzy rule learning through evolution for regression," Information Sciences, vol. 354, pp. 1–18, 2016.
26. L. Wang, Z. Mu, and H. Guo, "Fuzzy rule-based support vector regression system," Journal of Control Theory and Applications, vol. 3, no. 3, pp. 230–234, 2005.
27. R. Vaño, I. Lacalle, P. Sowiński, R. S-Julián, and C. E. Palau, "Cloud-Native Workload Orchestration at the Edge: A Deployment Review and Future Directions," Sensors, vol. 23, no. 4, pp. 2215, 2023. doi: 10.3390/s23042215.
28. Kubernetes, "Horizontal Pod Autoscaler." Kubernetes, kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/.

29. F. -D. Eyitemi and S. Reiff-Marganiec, "System Decomposition to Optimize Functionality Distribution in Microservices with Rule Based Approach," 2020 IEEE International Conference on Service Oriented Systems Engineering (SOSE), Oxford, UK, 2020, pp. 65-71, doi: 10.1109/SOSE49046.2020.00015.
30. N. Liu, Z. Li, J. Xu, et al., "A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning," in 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), IEEE, 2017, pp. 372–382.
31. E. Barrett, E. Howley, and J. Duggan, "Applying reinforcement learning towards automating resource allocation and application scalability in the cloud," Concurrency and Computation: Practice and Experience, vol. 25, no. 12, pp. 1656–1674, 2013.
32. Z. Ding, S. Wang and C. Jiang, "Kubernetes-Oriented Microservice Placement With Dynamic Resource Allocation," in IEEE Transactions on Cloud Computing, vol. 11, no. 2, pp. 1777-1793, 1 April-June 2023, doi: 10.1109/TCC.2022.3161900.
33. "aerOS project." https://aeros-project.eu (accessed May 25, 2024).
34. R. S-Julián, I. Lacalle, R. Vaño, F. Boronat, and C. E. Palau, "Self-* Capabilities of Cloud-Edge Nodes: A Research Review," Sensors, vol. 23, no. 6, pp. 2931, 2023. doi: 10.3390/s23062931.
35. X. He, H. Xu, X. Xu, Y. Chen and Z. Wang, "An Efficient Algorithm for Microservice Placement in Cloud-Edge Collaborative Computing Environment," in IEEE Transactions on Services Computing, doi: 10.1109/TSC.2024.3399650.
36. P. Ngatchou, A. Zarei, and A. El-Sharkawi, "Pareto multi objective optimization," in Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems, 2005, pp. 84–91. doi: 10.1109/ISAP.2005.1599245.
37. G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," arXiv preprint arXiv:1207.0580, 2012

Jorge Jiménez-García is telecommunications engineer with more than 5 years of research experience in the Biomedical Engineering Group (University of Valladolid, Spain). I have worked on the development of machine/deep learning algorithms to detect and assess the severity of various diseases. To achieve that goal, I gained expertise on image processing and analysis (retina/fundus images) as well as biomedical signal processing (cardio-respiratory signals). My research interests focus on AI applications in health care.

Ignacio Lacalle Úbeda (male) is a Researcher and Teaching Assistant at the Universitat Politècnica de València (UPV), a public University in the South-East of Spain. Ignacio is a Telecommunications Engineer (2014) from UPV and received his Ph.D. (Dr. Ing.) in 2022. The expertise of Ignacio is mainly rooted in the Internet of Things field, having participated in 10 national and EU research projects such as HE aerOS, MSCA AIAS, HE MISSION, H2020 ASSIST-IoT and others, related mainly to distributed systems, interoperability, added value services, data processing, cybersecurity and manageability, inter alia. Ignacio has performed various roles in those projects (ranging from Developer to Community Manager or Project Manager). In addition, most of those projects focused on applying IoT-related innovations, particularly in the fields of energy, maritime ports, manufacturing, agriculture, and others. Ignacio has authored or co-authored more than 30 publications in relevant journals and conferences.

Pawel Szmeja is an academic researcher from Polish Academy of Sciences. The author has contributed to research in topics: Interoperability & Semantic interoperability. The

author has an hindex of 13, co-authored more than 40 publications. Previous affiliations of Pawel Szmeja include Systems Research Institute & University of Warsaw.

Katarzyna Wasielewska-Michniewska is a researcher and practitioner with over 15 years of experience in scientific and commercial projects. Graduated from Warsaw University of Technology, Faculty of Mathematics and Information Technology, defended her PhD thesis in Systems Research Institute of Polish Academy of Sciences. Her initial scientific interests were agent programming, distributed systems, decision support and Internet of Things. With time, they evolved towards semantic technologies and (frugal/explainable) AI. Participant of H2020 projects: INTER-IoT, Assist-IoT, aerOS. She has published over 45 papers.

Przemysław Hołda, as a Computer Science student, was recognized as an award winner for "The Best Work in the field of Public Administration Digitization" by the Ministry of Digital Affairs and for "Beyond the Horizon" by the Polish Information Processing Society, in the "Miedzyuczelniany Konkurs Młodych Mistrzów 2022" contest. Some of his scientific activities were financially supported by a grant named "Najlepsi z Najlepszych 4.0" from the Ministry of Science and Higher Education. He attended the 4th Summer School on Cyber Physical Systems and Internet of Things (SS-CPSIoT'23) and the 23rd European Agent Systems Summer School (EASSS'23). He also had the opportunity to give guest lectures on Agent Systems and Applications at the Warsaw University of Technology. His current areas of interest are distributed systems and artificial intelligence.

Maria Ganzha is working as Associate Professor at the Warsaw University of Technology. She has received her M. A. and Ph. D. in mathematics at the Moscow State University in Russia, and a Doctor of Science degree in computer science from the Polish Academy of Sciences. Maria has published nearly 200 scientific articles, is editor in chief of a book series and a member of the editorial boards of 3 journals. She has been invited to the program committees of over 250 conferences. Her scientific interest is focused on agent-based technology, semantic technology, and machine learning. She is Technical Coordinator of EU project ASSIST-IoT, and principal investigator of SRIPAS team in another EU project aerOS. She is a President of Mazovia Branch of Polish Information Processing Society.

Carlos E. Palau Salvador is Full Professor in the Escuela Tecnica Superior de Ingenieros de Telecomunicación at the Universitat Politècnica de València. He has more than 20 years of experience in the ICT research area in the area of Networking. He has collaborated extensively in the RD of multimedia streaming, security, networking and wireless communications for government agencies, defence and European Commission. He has been the main UPVLC researcher in several funded projects, which has funded this work. He is author and co-author of more than 120 research papers and member of the TPC of several IEEE, ACM and IFIP conferences. He is Senior Member of IEEE. On the academical side he has been supervisor of 4 PhD Thesis, currently supervising three more and more than 100 MSc Thesis. Main researcher in TEMPUS and Asia Link projects with educational purposes.

Costin Bădică b. February 19, 1966, in Craiova, Dolj. He is a Professor Doctor Engineer at the University of Craiova, Faculty of Automation, Computers, and Electronics. Since

2012, he has been the Director of the ”Constantin Belea” Doctoral School within the Faculty of Automation, Computers, and Electronics at the University of Craiova. As of 2024, he is an Associate Member of the Academy of Technical Sciences of Romania (ASTR), Section: Information and Communications Technology, Computers, and Telecommunications. He serves as Co-Chair of the Technical Committee on Computational Collective Intelligence for the IEEE SMC Society. This committee received the ”Most Active SMC Technical Committee Award” in Cybernetics from the IEEE SMC Society in both 2021 and 2024. He is a member of the editorial boards for the following journals: Computer Science and Information Systems (ComSIS), ISSN 1820-0214; System Theory, Control and Computing Journal, ISSN: 2810–4099; Cybernetics and Information Technologies (CIT), The Journal of the Institute of Information and Communication Technologies of the Bulgarian Academy of Sciences, eISSN 1314-4081; Simulation Modelling Practice and Theory, ISSN 1569-190X (WoS); Vietnam Journal of Computer Science, ISSN (print): 2196-8888 — ISSN (online): 2196-8896. He is a founding member of the international conference IDC (Intelligent Distributed Systems) – 2007. His areas of interest include: artificial intelligence, multi-agent systems, distributed systems, and software engineering.

Stefka Fidanova is a Professor at the Institut of Information and Communication Technologies. The author has an hindex of 13, co-authored more than 240 publications.

Marcin Paprzycki works as an Associate Professor in the Systems Research Institute, Polish Academy of Sciences. The author has an hindex of 37, co-authored more than 750 publications.

Received: April 25, 2025; Accepted: December 22, 2025.