A Framework for the Operationalization of Analytic Workloads in Complex Distributed Computing Environments

Josu Díaz de Arcaya

Supervised by Dr. Aitor Almeida Escondrillas
and Dr. Ana I. Torre Bastida

Universidad de Deusto

This dissertation is submitted for the degree of
Doctor of Philosophy
is framed within the doctoral program
Engineering for the Information Society and Sustainable Development

April 2024

To my beloved family

Abstract

The use of AI-based technologies to improve business processes and competitiveness in an increasingly globalized market is on the rise. However, the success of these projects is still far from desired. In this thesis, we focus on various phases of the life cycle of machine learning technologies and develop technologies and applications that support professionals in increasing the success of these projects. For this, a culture of collaboration and communication within the organization is crucial, as is promoting skills training that goes beyond traditional software development. Moreover, organizations should focus on data lifecycle systematization. In this regard, the aim is to model the flow of the processes that make up the life cycle of machine learning applications through explicit representation. Therefore, one of the technical contributions of this thesis has been a domain-specific language that abstracts data scientists from the more technical aspects relating to the deployment and operationalization of artificial intelligence processes. This encourages these professionals to focus on obtaining business value from data while reducing the effort they must make in other areas of knowledge. Furthermore, other teams involved in the operationalization process gain greater clarity about machine learning processes, thereby increasing the efficiency of the project. On the other hand, the emergence of distributed computing paradigms such as cloud computing, combined with the enormous heterogeneity of devices on the edge of the network, makes even industry experts struggle during deployment. In this regard, we have developed a tool that uses genetic algorithms to optimize the deployment of machine learning flows based on opposing goals such as resilience, model performance, cost, and network performance. In addition, privacy and model performance criteria are considered. We have demonstrated that this tool achieves better results than experts in the field in all the objectives evaluated. Finally, this thesis opens various lines of research, such as the orchestration of services in 5G networks, the implementation of monitoring agents capable of anticipating and correcting problems, and the development of tools based on artificial intelligence in other phases of the life cycle, such as monitoring or training.

Resumen

El uso de tecnologías basadas en Inteligencia Artificial para la mejora de los procesos de negocio y la competitividad en un mercado cada vez más globalizado está en alza. Sin embargo, el éxito de estos proyectos está aún lejos de ser el deseado. En esta tesis ponemos el foco en diferentes fases del ciclo de vida de las tecnologías de aprendizaje automatizado, y desarrollamos tecnologías y aplicaciones que dan soporte a los profesionales de cara a aumentar el éxito de estos proyectos. Para ello, una cultura de colaboración y comunicación dentro la organización es crucial, así como promover la formación en habilidades que van más allá del desarrollo de software tradicional. Además, la sistematización del ciclo de vida del dato debe estar en el foco de las organizaciones. En este sentido, se apuesta por modelar el flujo de los procesos que componen el ciclo de vida de las aplicaciones de aprendizaje automático mediante una representación explicita. Por ello, una de las contribuciones técnicas de esta tesis ha sido un lenguaje de dominio específico que abstrae a los científicos de datos de los aspectos más técnicos relativos al despliegue y la operacionalización de los procesos de Inteligencia Artificial. De este modo, se promueve que estos profesionales se centren en obtener valor de negocio de los datos reduciendo los esfuerzos que deben realizar en otras áreas de conocimiento. Además, otros equipos que participan en el proceso de operacionalización consiguen mayor claridad sobre los procesos relativos al aprendizaje automático aumentando de este modo la eficiencia del proyecto. Por otra parte, la irrupción de paradigmas de computación distribuida como puede ser la computación en la nube, unida a la enorme heterogeneidad de los dispositivos en el borde de la red, hacen que incluso expertos del sector tengan dificultades durante la implantación. En este sentido, hemos realizado una herramienta que utiliza algoritmos genéticos para optimizar el despliegue de flujos de aprendizaje automatizado a partir de objetivos contrapuestos, como son la resiliencia, el rendimiento del modelo, el coste, y el rendimiento de la red. Además, se tienen en cuenta criterios de privacidad y rendimiento del modelo. Hemos demostrado que esta herramienta obtiene mejores resultados que expertos del dominio en todos los objetivos evaluados. Finalmente, esta tesis abre diversas líneas de investigación como la orquestación de servicios en redes 5G, la implementación de agentes de monitorización capaces de anticipar y corregir problemas, y el desarrollo de herramientas basadas en inteligencia artificial en otras fases del ciclo de vida como la monitorización o el entrenamiento.

Laburpena

Gero eta globalizatuago dagoen merkatu batean, gero eta handiagoa da adimen artifizialean oinarritutako teknologien erabilera, negozio-prozesuak eta lehiakortasuna hobetzeko. Hala ere, proiektu horien arrakasta ez da oraindik nahi bezalakoa. Tesi honetan, ikasketa automatizatuko teknologien bizi-zikloaren hainbat fasetan jartzen dugu arreta, eta profesionalei laguntzen dieten teknologiak eta aplikazioak garatzen ditugu proiektu horien arrakasta handitzeko. Horretarako, erakundearen barruan elkarlanerako eta komunikaziorako kultura funtsezkoa da, bai eta software tradizionalaren garapenetik haratago doazen trebetasunen inguruko prestakuntza sustatzea ere. Gainera, datuaren bizi-zikloaren sistematizazioak erakundeen fokuan egon behar du. Ildo horretan, ikaskuntza automatikoko aplikazioen bizi-zikloa osatzen duten prozesuen fluxua modelatzearen aldeko apustua egiten da, irudikapen esplizitu baten bidez. Horregatik, tesi honen ekarpen teknikoetako bat domeinu-lengoaia espezifiko bat izan da, datu-zientzialariak adimen artifizialeko prozesuen hedapenari eta operazionalizazioari buruzko alderdi teknikoenetatik bereizten dituena. Horrela, profesional horiek datuen negozio-balioa lortzean zentratzea sustatzen da, beste ezagutza-arlo batzuetan egin behar dituzten ahaleginak murriztuz. Gainera, eragiketa-prozesuan parte hartzen duten beste talde batzuek argitasun handiagoa lortzen dute ikaskuntza automatikoari buruzko prozesuei buruz, eta, horrela, proiektuaren eraginkortasuna handitzen da. Bestalde, konputazio banatuaren paradigmak, hala nola hodeiko konputazioa, sare-ertzeko gailuen heterogeneotasun izugarriarekin batera, sektoreko adituek ere zailtasunak izaten dituzte ezarpenean zehar. Ildo horretan, algoritmo genetikoak erabiltzen dituen tresna bat egin dugu, ikaskuntza-fluxu automatizatuen hedapena optimizatzeko, helburu kontrajarrietatik abiatuta, hala nola erresilientzia, ereduaren errendimendua, kostua eta sarearen errendimendua. Gainera, ereduaren pribatutasun- eta errendimendu-irizpideak hartzen dira kontuan. Frogatu dugu tresna honek adituek baino emaitza hobeak lortzen dituela ebaluatutako helburu guztietan. Azkenik, tesi honek hainbat ikerketa-ildo irekitzen ditu, hala nola 5G sareetako zerbitzuen orkestrazioa, arazoak aurreratu eta zuzentzeko gai diren monitorizazio-agenteen inplementazioa, eta adimen artifizialean oinarritutako tresnen garapena bizi-zikloaren beste fase batzuetan, hala nola monitorizazioan edo entrenamenduan.

Acknowledgements

We embarked on this endeavor together, since finding time for this meant less time for that. From the beginning, this was never possible without you, **Naiara**.

Oier taught me that even the tallest mountain becomes flatter once you start climbing, he is also the fastest runner in the family.

As **Laia** showed me, finding joy in mundane tasks makes them easier to bear. Nonetheless, she never backs down from a challenge.

I appreciate my **parents** relentlessly caring about my education, which more often than not must have required a lot of patience.

I can only be grateful to **Aitor Almeida** for having the ability to see the right direction at every step of this journey and always having a healthy sense of optimism about the outcome.

A special thanks to **Ana** for being the mastermind of this journey from the very beginning and for consistently making the difficult seem simple.

To all the **crew** in Tecnalia and Miñano, past and present, who have created a great working environment and helped me along the way. It is demonstrated that a tortilla-filled belly stimulates creativity.

I am grateful to my **friends** for helping me blow off some steam in the many situations one can face over the years.

To all the **people** I have encountered along the way, too many to name, who made me feel welcome wherever I have had the pleasure of being.

Table of contents

List of figures

List of tables

Acronyms

AI	Artificial Intelligence
AIOps	Artificial Intelligence for IT Operations
BDaaS	Big Data as a Service
CI/CD	Continuous Integration and Continuous Delivery
DataOps	Data and Operations
DevOps	Development and Operations
DSL	Domain-Specific Language
IaaS	Infrastructure as a Service
IaC	Infrastructure as Code
IoT	Internet of Things
ML	Machine Learning
MLaaS	Machine Learning as a Service
MLOps	Machine Learning and Operations
NFV	Network Functions Virtualization
PaaS	Platform as a Service
SaaS	Software as a Service
SDN	Software-defined networking
SLR	Systematic Literature Review

The secret of getting ahead is getting started.

Mark Twain

1

Introduction

THE adoption of artificial intelligence (AI) solutions is becoming ubiquitous, and organizations are increasingly advocating for including them in their processes to embrace the manifold benefits they entangle. Various techniques such as predictive maintenance, image recognition, natural language processing, fraud detection, and medical diagnosis have been widely applied in the industrial domain [52]. This trend has only been boosted by innovative paradigms such as Industry 4.0, which is also referred to as the fourth industrial revolution and builds on the concepts established by the previous one with the goal of raising global income levels and quality of life [249]. This revolution has only been possible due to the emergence of inexpensive intelligent infrastructural devices, the rise in data volumes, and improvements in connectivity. It integrates recent technologies such as the Internet of Things (IoT), AI, big data, and edge and cloud computing into the manufacturing sector.

Traditionally, organizations relied upon infrastructural devices located on their premises for their day-to-day operations, incurring significant upfront costs and making them responsible for hardware maintenance and updates. The cloud computing model has shifted the way organizations operate towards a more elastic hybrid cloud model, in which additional resources can be allocated as they are needed. Public cloud providers offer a range of services, such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a

Service (SaaS). Furthermore, the SaaS paradigm is continuously evolving and has led to the rise of a plethora of deeply specialized services in the fields of data science and data engineering, such as Machine Learning as a Service (MLaaS), and Big Data as a Service (BDaaS). On the other hand, edge computing promotes the use of distributed infrastructural devices closer to the data source, but at the expense of reduced computational power and heterogeneous architectures. This upsurge in innovative paradigms and services glimpses enormous potential but also carries an increased complexity that the various teams involved in the Machine Learning (ML) life cycle need to account for. To alleviate this, the data science, data engineering, and IT operations teams must join efforts, and the information flow between them needs to be strengthened. This is where innovative paradigms such as Machine Learning Operations (MLOps), which oversees the ML life cycle, and AI for IT Operations (AIOps), which applies AI technologies to the operationalization, can be beneficial.

The rest of this chapter provides an overview of the research effort undergone during this doctoral thesis. Section 1.1 provides the context and motivation that boosted this thesis. The hypothesis and objectives that were initially established are explained thoroughly in Section 1.2. Then, Section 1.3 describes the utilized methodology for achieving the goals. The contributions of this thesis are clearly identified in Section 1.4. Finally, the outline of this dissertation is described in Section 1.5.

1.1 Context and Motivation

Despite the increasing interest from companies to benefit from AI solutions, it has proven to be an elusive endeavor for decades [240], and many organizations are failing to obtain business value from their investments [33]. Data science projects tend to be complex by nature, as they involve various stakeholders within the organization with diverse backgrounds who are required to cooperate to obtain a common goal. Due to this, efficient communication is of paramount importance for the success of the project.

In addition, an in-depth understanding of the various phases that comprise the operationalization of AI is the cornerstone for the successful management of data science projects. In particular, the machine learning (ML) life cycle is described as the combination of various stages: data management, model learning, model verification, and model deployment [14]; however, these concepts only consider the AI dimension of the whole problem, and traditional software engineering practices should also be considered. During this research, the relevance of prototyping the application, continuous integration, and application monitoring stages is emphasized [147]. We reckon the importance of leveraging these two paradigms for AI operationalization and depict the combined life cycle in Figure 1.1. Next, we offer an in-depth description of these various stages:

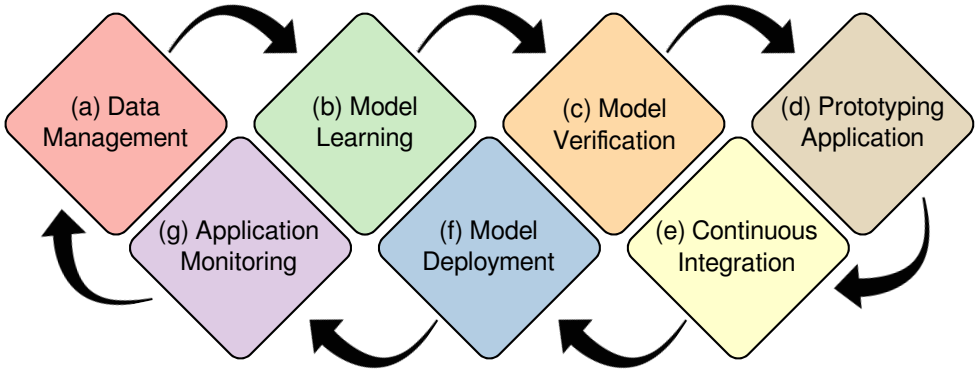

Fig. 1.1 The combined life cycle resulting from the combination of software engineering practices [147] applied to Machine Learning [14].

(a) *Data Management* is the preliminary stage towards the implementation of AI-based applications and emphasizes the need for organizations to adopt data governance principles and practices [200]. Among the many challenges that must be leveraged while implementing a data governance methodology, it is worth highlighting the following: i) data is often stored in silos across different departments and organizations [150], or even in various physical locations, hence the difficulty in accessing and combining it for its use in projects; ii) privacy is a major concern as the data utilized to train the algorithms tends to have sensitive information about the business [157]; iii) the data quality must meet the requirements of its intended use as it must contain all the necessary information, adhere to the rules and standards of the organization, and be free from errors and inaccuracies [112].

(b) *Model Learning* comprises selecting the model flavor [233] and utilizing the data to train the AI model. Some activities that typically fall into this phase are data preparation, model selection, hyperparameter tuning, and training itself [301].

(c) *Model Verification* is the stage in which the model is independently evaluated to assess whether it will continue to satisfy the requirements when exposed to new data [226]. It is an essential stage to guarantee that the model is accurate and reliable; model bias, explainability, and further concerns such as the overfitting or underfitting of the model are evaluated during this phase. At this point, the data science team has come a long way, and the results of the project should start to be tangible.

(d) *Prototyping Application* refers to the process of eliciting the technological requirements of the AI model that has been envisioned in previous stages, understanding the various data sources, the inner characteristics of the data, and the nature of the infrastructure in which the operationalization is taking place. It also oversees the harmonization with the security policies and applications already in place within the organization.

(e) *Continuous Integration* has become ubiquitous for building production ready software, as it has proven to be a game changer for raising the quality of software products. However, its adoption for ML endeavors remains cumbersome [159]. In this regard, already established methodologies such as unit testing during the pipeline do not directly fit the workflow of ML applications due to their static nature [261].

(f) *Model Deployment* encompasses the process of shipping the final product alongside its dependencies utilizing technologies such as containerization. The latest trends advocate towards also including the infrastructure by leveraging Infrastructure as Code (IaC) solutions.

(g) *Application Monitoring* refers to the continuous tracking and analysis of the performance and behavior of the AI solution, such as watching variables and metrics during the execution of the algorithms [30]. On top of the AI particularities such as model and data drift, and model bias; the infrastructural elements and software that runs on top needs to be scrutinized to prevent undesired downtime, and inaccuracies. This entire process is not linear, but rather a cycle that needs to be iterated to guarantee the robustness of the ecosystem.

The conceptualization of the aforementioned life cycle is only strengthened by the adoption of Cloud Continuum practices by organizations. In this paradigm, the computational requirements extend beyond the capabilities of on-premises datacenters, and extend towards using multiple entities such as the cloud, edge, and IoT; which provide additional analysis, processing, storage, and data generation capabilities [198]. The large heterogeneity of the various infrastructural devices that are found in this scenario escalates the complexity, and a streamlined methodology becomes necessary to navigate its nuances.

In this regard, software engineering teams have long adopted methodologies such as DevOps to automate software development, deployment, and maintenance. It is a set of practices and methodologies with the overarching goal of delivering software faster, with higher quality, and more reliability. Based on these principles, MLOps resides at the intersection of machine learning, DevOps, and data engineering [185], and is proposed as an efficient manner to incorporate ML models in production [265]. It correlates DevOps principles and methodologies to the ML ecosystem to raise the success rate of data science projects and

improve communication and collaboration between the data science and operation teams. On the other hand, AIOps refers to the process of leveraging AI and ML techniques to empower software and service engineers to operate online services and applications at scale [60]. It can be particularly beneficial in multivariate processes with a high degree of complexity, in which even experts in the field struggle to find near optimal solutions in a timely manner. Finally, Data and Operations (DataOps) is a recently coined term that aims to shorten the data analytic life cycle time by introducing automation in the data collection, validation, and verification processes [201].

Among the various reasons for which artificial intelligence projects struggle to provide the expected outcomes, it is worth highlighting that it requires a combination of skills ranging from data science, software engineering, to operations that is hard to find within organizations [61]. In addition, it is complicated to keep pace with the emergence of innovative technologies, paradigms, and solutions that are needed within the projects. In terms of the organization, a collaborative culture and communication need to be established, and changes in the processes need to be driven by the relevant business units. The data management stage of the ML life cycle displayed in Figure 1.1 is of paramount importance, and many organizations fail to accomplish even preliminary results due to a lack of a suitable process to handle their data resources [22]. Next, appropriate monitoring of the deployed AI assets is also a common pitfall, as it needs to leverage ML related features, such as biases and drifts that may occur over time [108].

In this context, we cannot but call attention to the importance of reinforcing the various stages within the ML life cycle with tools and methodologies that are able to facilitate AI adopters. Due to this, we deepen into the challenges and opportunities of data science projects, and provide tools that can aid in the operationalization of AI solutions in their different phases.

1.2 Hypothesis, Objectives, and Scope

Based on the current state of the art in tools and methodologies that support the various stages of the ML life cycle, the hypothesis of this dissertation is:

Hypothesis. *Using innovative Artificial Intelligence-driven tools and technologies to oversee the various stages of the machine learning lifecycle outperforms the results provided by experts in the field of Machine Learning operations.*

In order to validate the above hypothesis, the overarching goal of this thesis can be described as follows:

Goal. *To design and implement a domain-specific language (DSL) for the description of distributed analytical pipelines, which can be fed into a goal-driven operationalization framework for heterogeneous environments operated by complex metrics.*

This grand goal represents the direction towards which this dissertation is focused; however, it has been subdivided into the following more specific objectives to better measure their fulfillment.

O1 To provide insights into the challenges and benefits of the adoption of MLOps and AIOps methodologies in order to promote the adoption of AI-based solutions in both industry and academia.

O2 The analysis, design, and implementation of a description methodology for distributed analytical pipelines that oversees the definition, deployment, and monitoring of heterogeneous MLOps use cases.

O3 The design and implementation of a framework that can leverage conflicting objectives for the operationalization of distributed analytical pipelines.

O4 The design and implementation of an edge service that is able to evaluate heterogeneous infrastructural devices to obtain the necessary information for a near optimal deployment.

05 To state the conclusions of this dissertation and elicit future research inspired by the work undergone in this thesis.

An overview between the relationship of the various objectives described above and the different chapters of this dissertation is provided in Figure 1.3. The tools developed within this research are subject to the following requirements:

- Existing technologies and frameworks already in use by organizations should not be replaced unless strictly required. Instead, the integration with these systems should be seamless.
- Infrastructural devices are expected to be heterogeneous and distributed across the cloud and edge computational layers, hence it is important to leverage these challenges during the operationalization stage.

The work undergone during this research does not deal with the following conditions and challenges, which are considered to be out of the scope of this dissertation:

- The data distribution may change over time, leading to a phenomenon coined as concept drift, which results in a degraded performance of the ML model.
- Existing data biases are utilized to train ML models, which can perpetuate these biases, leading to unfair results.
- Explainability refers to the understanding of how the ML model internals operate and take decisions. Its relevance comes from the fact that transparency makes organizations more inclined to trust AI solutions.

It is worth noting that the overarching goal of this dissertation is to shed light on the various stages of the ML life cycle and to provide tools that aid data scientists, data engineers, and operation engineers in the operationalization of ML solutions.

1.3 Methodology

This section describes the research methodology that has been utilized in order to fulfill the goals and validate the hypothesis presented in Section 1.2. This methodology is highlighted in Figure 1.2 as a cyclic process, which is kicked off by a profound analysis of the existing literature and iterates over the various stages to obtain greater degrees of expertise in the research field.

- Exploratory Phase: the goal is to gather information by reading relevant articles from scientific journals and attending conferences aligned with the topic of this thesis. This knowledge paves the way to being able to grasp the gaps in the state of the art upon which the subsequent phases are sustained.
- Solution Design: once the research direction is revealed from the literature review, the study line needs to be defined in detail. The outcomes of the research in terms of concrete deliverables, the architecture of the tool, and the dissemination activities are typically planned during this stage. In this research, we identified that more effort was needed in terms of analytical pipeline definition, and that the elastic deployment of data science projects was not sufficiently studied.
- Implementation: the development of the frameworks that are going to be delivered as part of this research takes place during this stage. The DevOps methodology, which has also served as a building block in other parts of the research, has been applied to guarantee the quality of the software components.

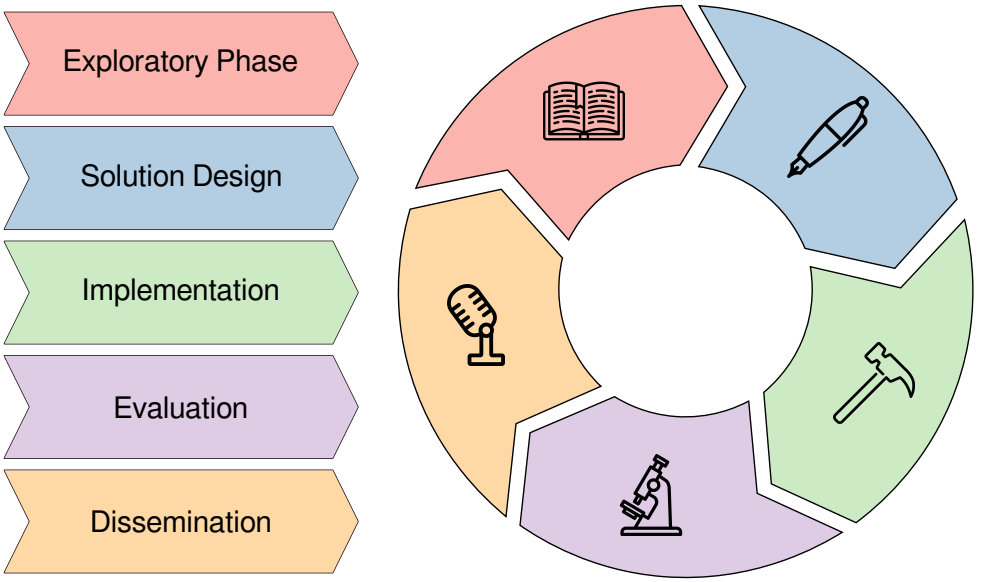

Fig. 1.2 Utilized research methodology.

- Evaluation: an appropriate assessment methodology needs to be planned beforehand as it will validate the quality of research. In this thesis use case validation and expert evaluation have been utilized. The experimentation has permitted us to highlight the advantages and limitations of the proposed research.
- Dissemination: various activities including scientific journal publications and conference papers have been delivered as part of this effort. Further details can be found in Section 1.4.

In spite of Figure 1.2 representing the research methodology as a linear process, there have been instances where a phase needed to be revisited more than once during the cycle (e.g., more research required in the exploratory phase). These dependencies between stages have been left out of the graph for simplicity. This cycle has been iterated several times during this research and has paved the way for the contributions described in Section 1.4. First, the exploratory phase has been particularly relevant for Chapter 2, in which a systematic literature review has been conducted on MLOps and AIOps methodologies. Next, solution design and implementation led the effort during Chapter 3 for the design of a Domain-Specific Language (DSL) for the analytical domain. Finally, in Chapter 4 the evaluation phase serves as validation for the work presented in this dissertation.

1.4 Main Contributions

The contributions of this dissertation can be organized into two categories: scientific and technical. First, the scientific contributions are the following:

SC1 An in-depth description stemming from the state of the art regarding the challenges and opportunities to consider for the adoption of MLOps and AIOps practices and methodologies, that aspires to boost the adoption of such methodologies in both industry and research. This contribution is detailed in Chapter 2.

SC2 An in-depth description derived from the state of the art regarding the frameworks, fields, and future trends in which MLOps and AIOps are thriving, that aspires to serve as a reference for future implementations regarding these methodologies. This scientific contribution is detailed in Chapter 2.

SC3 A schema for the conceptualization of data science pipelines that span across the edge and cloud computational layers. It provides means for the description, monitoring, and deployment of AI projects; and covers various stages of the ML life cycle. We present this in Chapter 3.

SC4 The mathematical formulation required for the operationalization optimization of data science projects. The following goals have been considered in this research; cost, network performance, model performance, and resilience have been modeled. This scientific contribution is developed in Chapter 4.

SC5 The future research directions stemming from the conclusions and the work presented in this dissertation. This contribution is presented in Chapter 5.

The following technical contributions are also the result of the work undergone during this thesis:

TC1 An application for the automated retrieval of scientific literature according to predefined queries that facilitate the process of doing a SLR. Chapter 2 contains the design and development of this contribution.

TC2 A DSL coined PADL for the conceptualization and operationalization of distributed analytical pipelines. It has been implemented utilizing JSON Schema [84], which facilitates its integration with various stages of the ML life cycle. This technical contribution is explained in detail in Chapter 3.

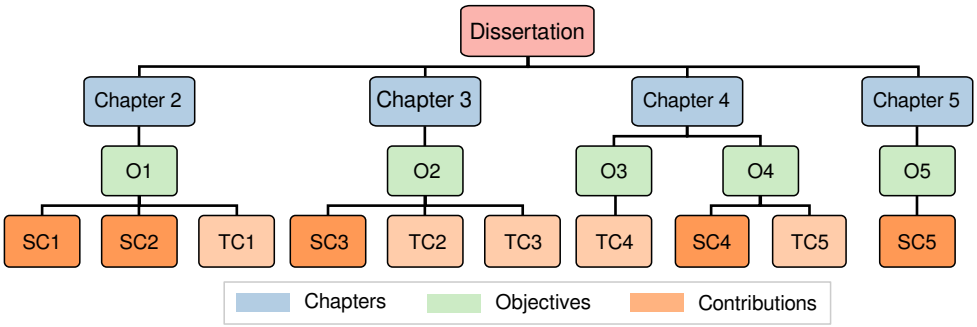

Fig. 1.3 The visual hierarchy of the chapters, the objectives, and the scientific and technical contributions for this dissertation.

TC3 A PADL online validator for early adopters to facilitate the adoption and modeling of existing analytical pipelines using PADL. In addition, a command line linter which permits IT Operations to directly tune their pipelines and integrate this technology into their operationalization processes. Details about this implementation can be found in Chapter 3.

TC4 A lightweight agent suitable for edge infrastructural devices coined metric shipper, which is not only able to retrieve fundamental metrics (e.g., CPU, memory), but also execute specific benchmarks specifically tailored for the data science domain (e.g., AI-Benchmark [140]). In addition, it has been developed to be cross-platform to be able to adapt to the heterogeneous infrastructural devices found at the edge. We showcase the agent in Chapter 4.

TC5 A pipeline orchestration framework coined Orfeon, which has been designed on top of the functionality offered by the PADL DSL. It provides means for the goal-driven optimization of data science pipelines across the various devices that can be found in the cloud and edge computational layers. All the details about the development can be found in Chapter 4.

All the technical contributions described above are publicly available on GitHub [77, 73, 75]. The objectives defined in Section 5.3 are each covered by a specific chapter of this dissertation. In addition, each objective has yielded certain contributions, both technical and scientific. This relationship between the chapters, the objectives, and the contributions is showcased in Figure 1.3.

1.5 Thesis Outline

This document comprises five chapters. First, Chapter 1 introduces the reader into the research field and continues by presenting the context and motivation for this research. Then, the hypothesis and objectives are presented alongside the research methodology that has been utilized throughout this thesis. Finally, the main contributions and their relationship with the various chapters of this thesis are showcased.

Chapter 2 examines the state of the art related to MLOps and AIOps methodologies and evaluates them against various research questions. In this regard, the challenges, opportunities, future trends, and architectures, are analyzed in this systematic literature review, and the conclusions provide relevant insights for the adoption of these methodologies in both industry and academia.

Chapter 3 explores the machine learning life cycle depicted in Figure 1.1. Then, based on related works, it elaborates a DSL coined PADL specifically tailored for the analytical domain that covers these very stages of the machine learning life cycle, ranging from data consumption to the monitoring stage. The overarching goal is to alleviate the burden of operationalizing data science projects in production environments, and bridge the gap between the data science, data engineering, and IT operations teams.

Chapter 4 builds on the foundations established by the previous chapters as it utilizes PADL to develop a novel framework for the optimization of distributed analytical pipelines. The optimization revolves around the resilience, cost, networking, and performance of the ML models. In addition, an edge-based agent is also provided to oversee the gathering of the necessary metrics for the optimization process.

Chapter 5 summarizes the work presented in this dissertation. The contributions achieved as part of this effort are outlined and serve as validation for the objectives and hypothesis presented in Chapter 1. Next, a list of relevant publications as regards international journals, conferences, and collaborations is presented. Finally, the future research areas that emerge from this dissertation are presented.

Nothing has such power to broaden the mind as the ability to investigate systematically and truly all that comes under thy observation in life.

Marcus Aurelius

2

Survey on MLOps and AIOps

The work presented in this chapter was published in 2023 in the ACM Computing Surveys journal [81], which was ranked as a Q1 in JCR. It received an impact factor of 16.6 in 2022 and was in the top three in the category "computer science, theory and methods". This chapter delivers SC1, SC2, TC1; therefore fulfilling O1.

Data science projects represent a greater challenge than software engineering for organizations pursuing their adoption. The diverse stakeholders involved emphasize the need for a collaborative culture in organizations. This article aims to offer joint insights into the role of MLOps and AIOps methodologies for raising the success of data science projects in various fields, ranging from pure research to more traditional industries. We analyze the open issues, opportunities, and future trends organizations face when implementing MLOps and AIOps. Then, the frameworks and architectures that promote these paradigms are presented, as are the different fields in which they are being utilized. This systematic review was conducted using an automated procedure that identified 44,903 records, which were filtered down to 93 studies. These articles are meant to better clarify the problem at hand and highlight the future areas in both research and industry in which MLOPs and AIOps are thriving. Our findings indicate that AIOps flourish in challenging circumstances like those presented by 5G and 6G technologies, whereas MLOps is more prevalent in traditional

industrial environments. The use of AIOps in certain stages of the ML lifecycle, such as deployment, remains underrepresented in scientific literature.

2.1 Introduction

Nowadays, artificial intelligence (AI) is increasingly commonplace, and disruptive AI solutions and developments are rapidly infiltrating comprehensively into multiple fields of human activities and domains. Beyond expectations, AI service development requires skilled professionals, quality labeled data, and rigorous development processes. In this context, it becomes necessary to highlight two crucial phases in the generation of AI-based solutions: the i) development of the AI solution as a software project, and the ii) accurate planning of the implementation and continuous deployment costs. Furthermore, AI models in production environments are a small part of a much larger ecosystem, the deployment and commissioning of AI models from a pure software perspective must be considered, and it is as important as the model itself. In addition, many experts share the vision that the development of AI-based solutions is equal to that of a pure software project. However, this is far from accurate since the development and operationalization of AI-based solutions have their own lifecycle, requirements, and particularities.

In the fields of software development and operations, good practices and methodologies have existed for a long time. The best known being the DevOps philosophy [90], an acronym formed by the words development and operations, which aspires to narrow the gap between these areas of expertise. Its goal is to accelerate the software development lifecycle while providing high-quality continuously delivered software components. This mature and widely adopted methodology encompasses and transcends agile software development and there exist multiple technological solutions implementing it. However; due to the rise of new technological paradigms such as Big Data, AI and 5G; DevOps methodologies need to be adapted to these new lifecycles and processes which expand beyond pure software solutions.

This gap materializes in one of the biggest challenges in AI practices today: the operationalization of AI solutions. Many organizations are desperate to figure out how to convert the insights discovered by data scientists into tangible value for their businesses, which has proven to be a cumbersome endeavor. It requires unifying multiple processes across diverse teams, starting with defining business goals and continuing all the way through data acquisition, and model development and deployment. This unification is achieved through a set of best practices for communication and collaboration between the data engineers who obtain the data, the data scientists who prepare the data and develop the model, and the operations professionals who serve the models. It is in this intersection where MLOps was conceived, aspiring to streamline the process of AI operationalization, whereas AIOps is

a viable solution to the upsurge of growing IT infrastructures and data. Both are the main exponents of an explosion of Ops variants related to AI, as they can be LAOps [204] or RLOps [172]. Therefore, we focus this survey on them two, instead of extending it to other methodologies that have not yet proved to be of interest to organizations.

The objective of this systematic literature review (SLR) is precisely to analyze the challenges and opportunities of these methodologies, in order to promote their benefits and adoption in both industry and academia. The expected result of this work is that those who read it can advance in their different domains and areas of expertise, due to being clearer about the main concepts of MLOps standardization, and the possibilities of providing intelligence and automation to their processes through AIOps. Finally, the various frameworks and domains in which MLOps and AIOps are thriving are shown in detail.

2.1.1 Comparison between MLOps and AIOps

Due to the fact that both disciplines are relatively young and their definitions differ depending on the source, it is challenging to establish a comparison between them. Furthermore, in spite of their substantial differences, there are some overlaps in the teams, skills, and challenges required to successfully implement them. First, MLOps is a well-established term in industry, although it was first mentioned in scientific literature in 2020, a contemporary definition is provided by the authors of [166], in which the authors argue that MLOps leverage machine learning, DevOps, and data engineering, with the goal is to productionizing machine learning systems by facilitating the creation of machine learning products. It stands on principles such as CI/CD, collaboration, orchestration, reproducibility (data, model, and code versioning), and continuous monitoring. The use cases in which the MLOps paradigms is applied vary greatly, from failure prediction, self-driving vehicles, and defect detection [154] to the electricity market [263]. On the other hand, the first mention of AIOps in the scientific literature dates back to 2019. The authors in [239] argue there is no generally accepted definition of AIOps in the scientific literature and highlight the importance of artificial intelligence to create self-learning and self-healing applications and infrastructure in the field of IT. Furthermore, AIOps is regarded as the only viable solution to deal with the expanding IT infrastructure, according to the findings of [48]. In this regard, observability is crucial for the adoption of AI-based solutions, and is being conducted in various areas, including distributed cloud applications [278]. Due to the scarcity of manuscripts on AIOps in the scientific literature, we have explored the gray literature and how the major providers in the market approach AIOps to gain a better understanding of the evolution of this paradigm. IBM provides various AIOps solutions for the following case studies: scaling performance and sustainability, optimizing IT environments, maximizing uptime, and managing spikes in

	MLOps	**AIOps**
Inception	2020	2019
Concept	Machine Learning Operations.	AI for IT Operations.
Goal	Creating stable products from ML prototypes.	Cope with the growing complexities posed by modern IT systems, reduce human intervention.
Enablers	CI/CD, orchestration, reproducibility, collaboration, continuous monitoring.	Big Data, cloud computing, observability.
Focus	Multidisciplinary.	Information Technology.
Use Cases	Failure prediction, autonomous vehicles, defect detection, electricity	Spikes in demand, maximize uptime, optimize environments, improve efficiency.

Table 2.1 Comparison framework between the MLOps and AIOps paradigms

demand [136]. For AWS, the goal of AIOps is to reduce human intervention in IT operations processes. Microsoft claims that AIOps helps achieve high quality and efficiency with less human intervention [192]. In this study by Forrester, commissioned by Google [54], they elaborate that AIOps is key to solving cloud operation challenges and improving efficiency and productivity.

On the basis on these findings, a comparative framework between MLOps and AIOps is proposed in Table 2.1, which aims to serve as the stepping stone for the reader to dive into this survey. It is noteworthy to mention that while the first appearance of these two terms in scientific literature is very recent, earlier research has examined the challenges posed by them.

2.1.2 Objectives

This SLR is framed in the fields of MLOps and AIOps. The motivation for doing this SLR are: to determine the causes that are preventing these fields from having a more widespread adoption in industry and academia, to understand how to overcome these issues by revealing the opportunities that are currently arising for MLOps and AIOps, to discover the architectures and frameworks that may aid to embrace these methodologies, and to have a clear picture of which are the current and future areas in which these methodologies are being applied. Due to this, in Table 2.2 we have elaborated a set of research questions and their purpose.

The rest of this paper is structured as follows: Section 2.2 covers the methodology utilized in this systematic literature review, diving into the details about how the manuscripts have

	Question	Motivation
Q1	What are the open issues, challenges and particularities in MLOps and AIOps?	To have a clear view of the inherent difficulties of embracing MLOps and AIOps, so adopters can prepare accordingly.
Q2	What are the opportunities and future trends in MLOps?	To understand which are the areas that may be interesting to explore in the field of MLOps.
Q3	What are the opportunities and future trends in AIOps?	To understand which are the areas that may be interesting to explore in the field of AIOps.
Q4	What frameworks and architectures facilitate MLOps and AIOps?	To have a better understanding of how industry and academia are utilizing MLOps and AIOps from a practical standpoint.
Q5	What are the current and future fields in which MLOps and AIOps are thriving?	To determine the areas in which MLOps and AIOps are being applied and considered.

Table 2.2 Research questions and their motivations.

been retrieved and selected. An overview of the selected studies can be found in Section 2.3. In Section 2.4 the studies that have been included in the review are used to offer insights into the selected research questions displayed in Table 2.2. Finally, the conclusions of this research are summarized in Section 2.6.

2.2 Methods

In [160] the authors summarize the required phases for performing a SLR, which we have taken as a reference in this manuscript. The research questions are specified, and a review protocol is developed during the planning phase. Next, the primary studies that are to be included in the review and a quality assessment on them is performed during the conducting phase. Finally, the formatting of the main report takes place in the reporting phase.

2.2.1 Article retrieval

Automatic retrieval of manuscripts provides a more comprehensive, less error-prone, and unbiased search. To do this, we have considered and tested various repositories, such as arXiv [276], Springer [260], and IEEE [138]. However, not all the necessary features were available, and others, such as MDPI and ACM do not expose their databases through

dedicated services. This is when initiatives like Crossref [57] come in handy because they provide a unified point of access for the published manuscripts of the various editorials; hence, only a single client is necessary for performing the automated search, which streamlines the development process. On the downside, not all the editorials deposit all the metadata for their manuscripts; vital information used in this SLR, such as the paper abstract, is not always available. Finally, Elsevier [36] through its renowned Scopus service, provides metadata search for a wide variety of editorials; abstract information is available; and its API is feature rich. For this reason, we have developed a client service that interacts with the Scopus API to gather all the manuscript metadata that has been used in the selection process. The source code for this client, along with the documentation for its use, is publicly available on GitHub [76].

2.2.2 Search terms

The search process has been the result of the combination of concepts in the different columns defined in Figure 2.1. Each of them has been selected by the authors based on their expertise in the field. The blue column stands for the main topic and situates the search in the right field. The purpose of the red column is to narrow the search towards actions addressing MLOps, depicted in a darker shade of red, or AIOps, depicted in a lighter shade of red. Additionally, some actions pertinent to both fields contain both colors. Finally, the green column is the dimension, and it serves the purpose of providing a better alignment of the search string with the research questions outlined above. Following this method, three hundred search queries are constructed, and we utilize the search capabilities provided by Scopus to find articles that contain all three terms of the search query in the title, abstract, or keywords. The search has been narrowed to articles published between 2018 and 2023. One of the benefits of performing an automated search of scientific databases is that the searched space is considerably wider than that of a manual approach. For this reason, the three separate groups that cover topic, action, and dimension make up for six hundred different search strings.

While the search procedure described above reveals a vast array of studies, works about AIOps are significantly underrepresented in the scientific literature in comparison to MLOps. Due to this, yet another query specifically tailored to discover those studies addressing AIOPs has been applied. The authors chose the search terms for this procedure using both their professional knowledge and relevant scientific literature [48, 206]. Figure 2.2 depicts the search string utilized to discover high-quality studies in the field of AIOps.

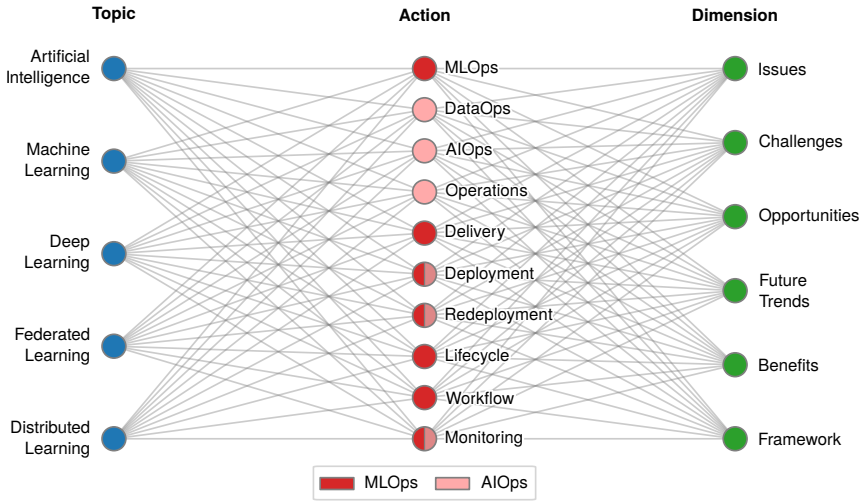

Fig. 2.1 Search queries utilized to identify the manuscripts that are included in the SLR.

```
'AIOps' AND ( 'Failure Prevention' OR 'Failure Prediction' OR
'Failure Detection' OR 'Root Cause Analysis' OR 'Remediation' OR
'Resource Consolidation' OR 'Scheduling' OR 'Power Management' OR
'Service Composition' OR 'Workload Estimation' OR
'Event Correlation' OR 'Failure Mitigation' )
```

Fig. 2.2 Search string utilized to identify high-quality studies on AIOps.

2.2.3 Selection Criteria

Table 2.3 summarizes the inclusion and exclusion criteria that has been applied in this systematic review to narrow down the number of articles and analyze the most relevant ones for the topic in hand. Considering that both MLOps and AIOps are recent concepts, and the upsurge of new paradigms such as cloud or edge, which introduce new concepts in the field of AI operationalization, only studies from 2018 onward have been considered.

Therefore, the list of articles included must have been identified in the automatic search and hold interesting or groundbreaking ideas aligned with the topic of the review. On the other hand, articles not written in English and retracted publications are not included. In addition, studies being released by publishers clearly not aligned with the main topic are excluded. We have made every effort to obtain the articles with the subscriptions available by the authors. However, any article behind a paywall or unavailable becomes excluded as well.

	Inclusion Criteria
I1	Published between 2018 and 2023.
I2	Identified by the search queries.
I3	Articles presenting new interesting ideas.
I4	Manuscripts closely aligned with the topic.
	Exclusion Criteria
E1	Publication not in English.
E2	Retracted publications.
E3	Publisher not aligned with the study.
E4	Publications behind a paywall that cannot be retrieved.
E5	Articles with insufficient citations.

Table 2.3 Study selection criteria.

Finally, an automated minimum threshold based on the number of citations is applied to the articles. Manuscripts from 2022 onward are not filtered as we understand this would leave out interesting research that has not yet reached a wider audience; articles from 2021 must be cited at least once, two cites for articles from 2020, three for 2019, and four for 2018. This filter promotes the inclusion of newer ideas and preserves the freshness of this systematic review.

2.2.4 Quality Metrics

One of the defining features of a systematic review is to give an appraisal of the quality of the studies included [12]. To this end, in Table 2.4 we have listed a set of quality metrics to measure the included manuscripts and rank them. These metrics have been gathered from various sources, such as PRISMA [219, 220], other literature reviews [17, 277], and our expertise on the field.

After evaluating these metrics upon the included articles, each of them ends up with a total score between 0 and 10. We classify them as deficient (0-2), sufficient (3-4), good (5-6), very good (7-8), and excellent (9-10). Please note that the mark does not directly relate to the quality of the article, instead it provides an approximation of its alignment with the goals of this SLR.

2.3 Results

This section is devoted to supplying an overview of the selected studies. In Subsection 2.3.1 the methodology for the retrieval and selection of the manuscripts included in this

	Quality Metrics	Value	Weight
M1	It provides a comprehensive state of the art aligned with this SLR, and identifies the knowledge gaps to justify the reason of the study.	0/1	1
M2	The research is validated against at least one use case aligned with the objectives of this SLR.	0/1	2
M3	The number of research questions that are tightly coupled with the manuscript.	[0..5]	2
M4	The manuscript has been published under an open license.	0/1	1
M5	Publication type (Other / Journal).	0/1	1
M6	The manuscript has been published in a journal in the first JIF quartile.	0/1	1
M7	Citation count.	[0..3]	2

Table 2.4 Quality metrics for evaluating the articles.

SLR is offered. Next, the risk of bias and validity threats are explained in Subsection 2.3.2. Finally, Subsection 2.3.3 discusses a high-level overview of the selected studies.

2.3.1 Study Selection

The PRISMA statement [219, 220] emphasizes the need to describe the results of the search and selection process potentially by using a flow diagram. In [119], the authors highlight again the importance of such a diagram in systematic reviews to facilitate the reader a rapid understanding of the core procedures utilized. In addition, they also provide an application which we have used to generate the diagram in Figure 2.3. Two phases have been conducted to obtain the final number of articles included in this systematic review: an identification phase, and a screening phase. After which the remaining studies are selected for further analysis.

As mentioned in the previous section, the search process, retrieval, and storage of the metadata associated with the studies has been made programmatically. Due to this, a daunting total of 44,903 articles have been identified by the search queries detailed in Table 2.1. It is worth noting that duplicated records were already filtered during the retrieval process, hence are not included in this number. Then, a series of filters have been applied before the screening phase. there are 453 records for which no title has been obtained, 9 retracted articles, 36 erratum articles, the publisher not aligned with the field of the study has resulted in 12,863 manuscripts being excluded, 1,057 full conference proceedings are excluded as we understand the meaningful articles should have been already identified in this research. Finally, we have applied an automated filter to prioritize recent articles over older ones based

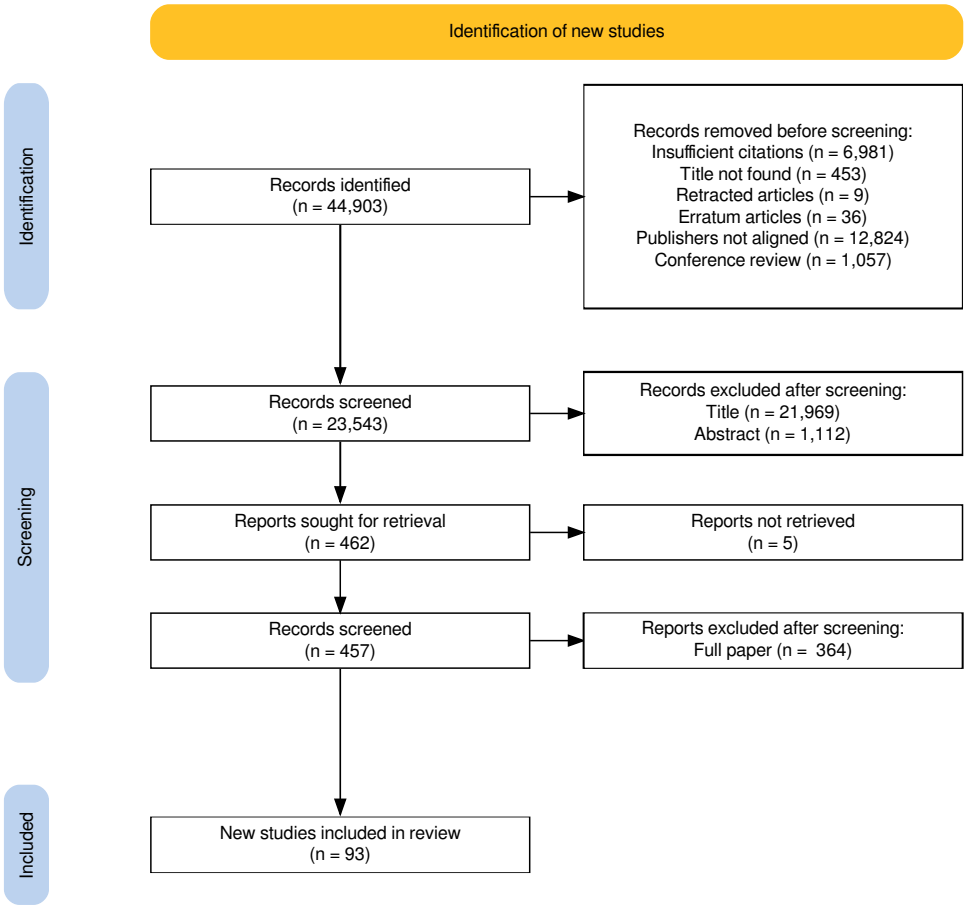

Fig. 2.3 Flow chart showing study selection.

on the citation count. This results in 23,508 being handed over to the screening phase, in which the inclusion and exclusion criteria detailed in Table 2.3 is to be applied. This results in 21,940 studies being excluded due to their title, and 1,111 due to their abstract. Up to this point, the analysis has been performed purely on the metadata being retrieved from Scopus by the automated search client. The remaining 452 papers have been downloaded for deeper analysis, after screening the whole paper the authors have selected 93 articles for further analysis in this systematic review.

Table 2.5 showcases the various stages executed during the selection process to obtain the included studies. During stage 1, we have solely used Scopus which takes away the nuances between the different databases and their APIs (e.g., some databases do not provide

Stage	Process	Selection Criteria	ACM	Elsevier	Springer	IEEE	MDPI	Others	**Total**
1	Identification	Search terms	1,612	6,666	5,729	11,959	3,463	15,474	**44,903**
2	Identification	Filtering	1,064	6,121	3,961	8,316	3,098	963	**23,543**
3	Screening	Title	115	344	287	618	158	52	**1,574**
4	Screening	Abstract	42	85	87	192	37	19	**462**
5	Screening	Full paper	10	9	10	50	7	7	**93**

Table 2.5 The manuscripts filtered at the various stages broken down by editorial

the possibility of performing complex queries, others do not offer an API, some establish a threshold on the maximum queries). We deem this appropriate as Scopus integrates a large database of the most relevant publishers in the field of this SLR. Stage 2 has been performed entirely by directly applying the filters on the database itself. Stages three and four have been executed by the authors by showing merely the information required for that particular stage (i.e., the title in stage 3, and the abstract in stage 4). Finally, a full reading of the article has been done in the final stage, which results in the studies finally included as part of this review. These researches are then fully read and ranked against the metrics defined in Table 2.4, the result of this work is showcased in Table A.1.

During the final screening phase, there are various manuscripts that even though they appear to comply with the criteria in Table 2.3 they were finally not included in this SLR. In [311] the authors provide a survey on ML testing, and outline research trends and challenges in the field. A DNN tuning framework on mobile frameworks is presented in [299]. In [50] the authors propose a framework to provide forecasting both at the edge and cloud layers. Future directions at the intersection of machine learning and operations management are highlighted in [21]. Even though we found these manuscripts fascinating they were not included as part of this SLR due to the impossibility to fit their findings within the research questions described in Table 2.2. However, we cite them because we reckon they mention insightful ideas and may be suited for similar researches.

2.3.2 Risk of bias

The fact that the initial search process has been performed programmatically diminishes the risk of the researcher bias in the selection of the studies. Even though we acknowledge the chance of relevant manuscript not being identified by the automated process, the sheer amount of studies identified by the wide search queries outlined in Figure 2.1 constitute sufficient research material for this SLR. In addition, the reason for not having included non-academic sources such as gray literature into this study is that this research does not suffer from low volume nor quality of evidence [102]. Finally, the selected subset of studies has

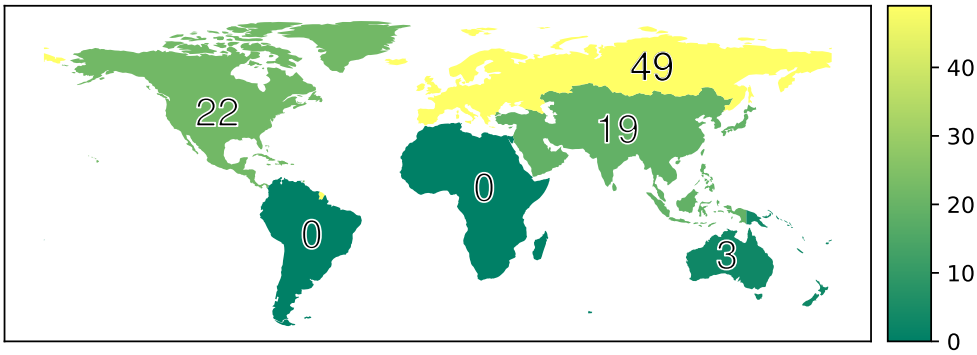

Fig. 2.4 Geographical distribution of the selected studies.

been qualitatively analyzed following the criteria in Table 2.4, for which we have positively valued quantitative statements. Furthermore, several rounds have been performed by the authors of this study during the various stages to minimize personal preferences and biases.

2.3.3 Overview

In terms of geographical distribution, a significant amount of the selected papers have been published in Europe ($\approx$ 51 %) as shown in Figure 2.4, followed by North America ($\approx$ 23 %) and Asia ($\approx$ 20 %). Only three papers out of the 93 have been finally selected in Oceania. Unfortunately, no manuscript from South America nor Africa has reached the final stage of selected papers. We reckon that a plausible explanation for this distribution is that considering that all the authors of this systematic review reside in Europe, the search queries defined in Figure 2.1 have a closer alignment with the research promoted in this continent. Next, MLOps, AIOps, and AI operationalization are in fact hot topics in North America, Europe, and Asia.

With regards to the nature of the selected manuscripts, the majority of them are conference papers ($\approx$57%), followed by journal articles ($\approx$41%). This distribution is pertinent to this research since one of the quality metrics defined in Table 2.4 is in fact the publication type, for which we favor research published in journals as they tend to be longer pieces of work with more insightful results.

The data presented in Figure 2.5 is broken down by year and includes two pieces of information: the i) number of manuscripts included in this research aggregated by publisher represented in the left-hand vertical axis, and the ii) total records initially identified showcased in the right-hand vertical axis. The former shows an increasing trend in the interest in the topics discussed as part of this research in recent years, with a steady upwards slope between

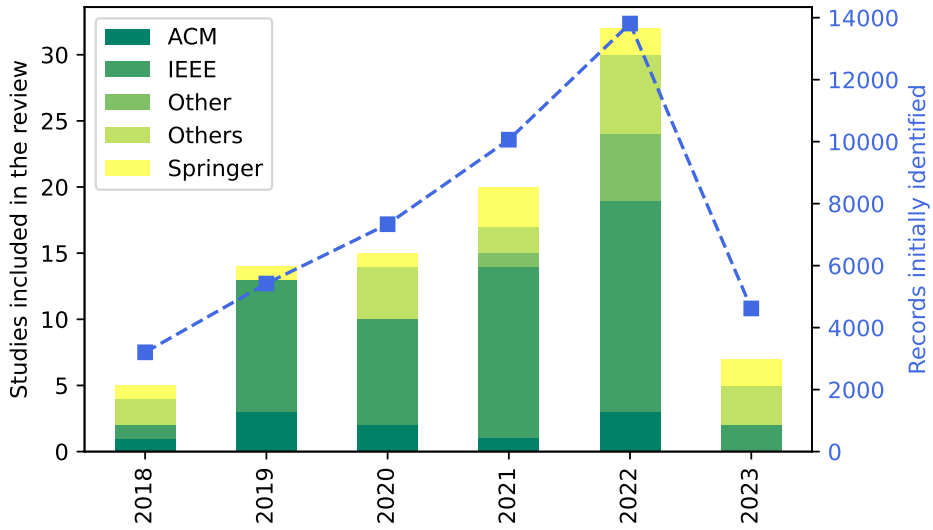

Fig. 2.5 Cumulative number of manuscripts by year and publisher.

2018 and 2020, and a milder slope between 2020 and 2022. The latter offers a higher granularity on the number of articles published by the different editorials over the years.

Finally, in Table A.1 there are five manuscripts that stand out from the rest after evaluating the quality metrics defined in Table 2.4. The authors in [59, 149] circumscribe their work to AI operationalization in 5G and 6G environments, respectively. Similarly, the deployment of ML solutions for network traffic classification is described in [218]. In [34, 289] the authors introduce their frameworks for the deployment and redeployment of smart algorithms in production systems.

2.4 Discussion

This section thoroughly examines the studies included in this review and classify their insights against the research questions described in Table 2.2. The open issues and challenges of MLOps and AIOps are discussed in Subsection 2.4.1. Next, the opportunities and future trends of MLOps and AIOps are outlined in Subsection 2.4.2, and Subsection 2.4.3 respectively. In Subsection 2.4.4, different frameworks and architectures for both paradigms can be found. Finally, in Subsection 2.4.5 the fields in which MLOps and AIOps thrive are described.

2.4.1 What are the open issues, challenges, and particularities in MLOps and AIOps? (RQ1)

Due to the relative youth of these two paradigms, scientific literature concerning their issues and challenges exhibits many similarities. For this reason, this section provides insights from a joint perspective. The implementation of MLOps and AIOps based projects in production environments is cumbersome, and a combination of cross-domain skills and a collaborative culture is necessary to accomplish such an endeavor [40, 258]. One of the main problems is that data scientists are unfamiliar with the unique requirements and characteristics of some environments [32] and the level of understanding they must acquire in order to build a solution that includes business value, data, system, and process integration considerations [60], and infrastructure [183]. Unlike in research, industry requires a balance between accuracy and processing, and systems tend to be more complex [197]. Next, software developers struggle to acquire ML skills due to its interdisciplinary nature [5] and their lack of ability in feature engineering, parameter tuning, and model selection; combined with the myriad ML libraries and frameworks [25]. Finally, the complexity of ML workflows forces operations engineers to have a high degree of application and platform expertise to size, provision, and operate the required resources [110, 43]. In summary, this complexity is stemmed by the difficulty to find within teams the large variety of skills required to operationalize data science projects [10, 179], and the different backgrounds and knowledge of the involved stakeholders [316, 15, 155, 239]; hence the high value of having a strong background in both software engineering and data science [118] for the project to succeed.

Computer Engineering

On the software side, a more streamlined and systematic approach to AI application development and lifecycle management is needed [133], and the interaction between software engineering and ML workflow activities needs to be better reported [180]. However, data scientists are not necessarily computer engineers by training [266], which threatens the maintenance and evolution of AI solutions. In fact, they often fail to understand the different frameworks utilized during the development process [115] and the unique requirements and characteristics of certain environments [32]. In this regard, experienced IT professionals still question the efficiency of ML-based solutions [239], partly due to the lack of explainability of such systems [2]. In addition, the patterns and practices used in software engineering do not fit those required in data science [60] and introduce the challenges of data availability, concept and data drift [151], and scalability [253], leading to failures and reduced model accuracy in ML deployments [221]. For instance, the complexity of AI designs represents a bigger challenge [92], such as in terms of modularization [10], than traditional software

components and requires a working understanding of ML principles and proficient technical expertise [107]. In addition, common frameworks utilized in software engineering, such as source code versioning tools, fail short when being used in data science projects due to their close relationship with the associated data [179]. For instance, in spite of the popularity of GitHub as a hosting platform, there is a lack of open-source ML-enabled projects leveraging GitHub Actions [38]. In addition, version control management, metainformation definition, and data and model governance need to be addressed [309].

Data Management

The importance of data management cannot be undermined in both methodologies. Aspects such as data quality [183, 239], data access [183, 29, 118], data preparation and labeling [179, 5, 297], data validation and cleaning [231], limited availability of datasets [2], or having to integrate the data from different sources [316, 182]; are often overlooked and may consume a large part of the project's time and budget [179, 118] and may require the involvement of experts in the field [297]. In addition, not only is there a tight dependency between the developed models and the associated data [223], but also the data discovery, management, and versioning are more complex in ML-based scenarios [10, 179]. The Big Data paradigm [7, 60, 153, 25, 99] adds yet another layer of complexity to data management in ML projects, aspects such as task distribution and data movement [153], or batch processing [25], become even more relevant in these scenarios. In addition, the underlying complexity of ML and Big Data workflows resorts to time-consuming, convoluted coding [7]. Due to this, the joint work of industry and academia is beneficial in these endeavors [60].

Orchestrating the ML Lifecycle

An efficient distribution, parallelization, and orchestration of the data and tasks of an ML solution are of paramount importance for the success of any project [209, 153]. In this regard, innovative technologies for the execution of distributed tasks, such as serverless, promote the use of multistage machine learning solutions by guaranteeing optimal response and running time [16]. On the other hand, opting for a distributed deployment requires dealing with the traffic overhead, bandwidth, and latency delays of the deployed services [304], as well as the scalability of the computation resources associated with the deployment [31, 218]. These pros and cons are particularly relevant when utilizing different computational layers for the deployment of data science projects. For instance, metaheuristic techniques can be used in cloud computing deployments to maximize the degree of isolation and resource sharing among the components [211]. In addition, cloud deployments can be extended to the edge of the network, but the communications (e.g., bandwidth, latency, connectivity) [126,

304, 294, 92, 308], scarce computing resources [126, 99, 304, 115, 272, 171], storage space [272], memory [99, 304, 189, 115], security [106], privacy [106, 308], and energy consumption [189, 294, 126, 115, 209, 189, 308] where GPU dependent applications tend to be particularly energy-hungry [92]; need to be thoroughly scrutinized in order to achieve the best performance. The use of edge computing and IoT [163] technologies keeps rising, but represents a challenge due to the restrained resources found in this computational layer, hence they need to be used efficiently [126]. In addition, protecting the intellectual property and integrity of the deployed models is complex due to the distributed nature of edge devices. The aforementioned drawbacks lead to data often having to be transferred over to more resourceful infrastructural devices [126]. In addition, edge nodes often operate in unreliable environments [169] and need to undergo cumbersome manual optimizations [34]. Containerized technologies are often utilized for delivering the ML solution, but more emphasis is required on dependency and filesystem management [216]. In general, the lifecycle of data science projects must leverage the following aspects: i) ML workflows often operate in unreliable, harsh, and constrained environments [169, 286, 209, 296], and ii) the trade-off between the benefits of using high computational resources and more humble architectures [289].

Hardware and Architectures

Another key aspect to consider in MLOps and AIOps is the infrastructural devices available during the various stages of the ML lifecycle, where an appropriate configuration of the hardware represents a challenge in DNN solutions [264, 34]. The large heterogeneity of the hardware platforms [25], their diverse characteristics [282], and restrictions [296] such as limited performance or memory [99], makes hardware configuration an important task to consider in every data science project. On the other hand, the use of technical software [31], incompatibilities between ML libraries [305], and the different architectures and resources imposed by the models [283]; pose yet another threat to the materialization of the project. In addition, yet another challenge is the differences in architecture designs and implementation of the various framework found in the development and deployment of the data science solution [115]. The security and reliability of the ML workflow needs to be assessed from a data perspective [197], and from the degree of isolation of the different components forming it [211]. In the past, computational power has slowed down the development of Large Language Models (LLMs) and conversational agents, whereas now the challenges are more related to the trustworthiness of the results [248].

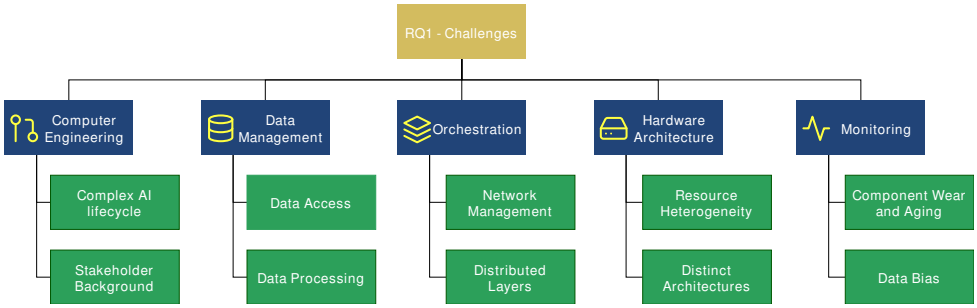

Fig. 2.6 A summary of the challenges associated with the adoption of MLOps and AIOPs methodologies classified by category.

Monitoring

The different components that comprise data science projects require continuous monitoring in order to detect deviations during their runtime [179, 129]; hence the use of dedicated tools is mandatory [137, 197]. The static nature of trained ML models severely affects dynamically changing environments [235]. Due to this, ML models must be responsive to changes such as component wear and aging [223], and data biases [197] which can result in models becoming outdated due to concept drift [230]. However, the definition of performance metrics to monitor the deployment is problem-specific [118], and some authors propose a method for inferring application KPI degradation, without having to wade into application-specific metrics, for application engineers to rely on [110]. The recent upsurge of LLMs raises several concerns. In terms of fairness, AI applications at various stages of development may be biased by using training data in dubious ways. Mental health risks are associated with human participation in the MLOps cycle, which has implications for transparency, and the use of frameworks for developing reliable generative AI applications is crucial for explainability [121].

To top it all off, data science endeavors very often suffer from unrealistic expectations [183], and business, architectural, process, and organizational challenges are to be expected while adopting ML-based solutions in traditional industries [87], which in conjunction with the aforementioned challenges results in a degraded productivity of the stakeholders due to the interdisciplinary nature of the projects and management complexity [43]. The summary of the main areas of interest related to this research question is depicted in Figure 2.6.

Implementing MLOps and AIOps is cumbersome and requires a combination of cross-functional skills and a collaborative culture within the organization. To be successful, projects require a combination of software engineering, data science, and operations expertise that is difficult to find. The importance of proper data management is paramount in data science projects, which is only exacerbated in big data ecosystems. Tools and frameworks for efficient distribution, parallelization, and orchestration of data and tasks are necessary due to the different computing tiers (i.e., cloud, edge) in which these projects operate, but also present multiple challenges (e.g., scarce computing resources, security, privacy). In addition, these environments often consist of heterogeneous infrastructure devices with different characteristics and requirements. Next, monitoring the production environment promotes the responsiveness of such models to the various drifts. The scientific literature on this topic is in a state of ongoing development, and more extensive research will be required in the future to identify distinct challenges for both methodologies.

2.4.2 What are the opportunities and future trends in MLOps? (RQ2)

This subsection provides an analysis and categorization of the various studies that address the potential applications and emerging trends of MLOps studies. Figure 2.7 depicts the primary areas on which academic studies are focused. The cloud continuum and edge computing, followed by harnessing the AI lifecycle, are the main opportunities identified by researchers.

Industry

There is a need for accurately measuring the business impact of AI solutions on businesses [87]. Data science projects require the involvement of different business layers to succeed, and leadership, executives, and stakeholders need to be on board with the MLOps strategy [40]. In this regard, education and training are required for ML activities [40, 10]. The use of ML algorithms has shown enormous potential for complex critical systems and processes [7, 180].

AI Lifecycle

In order to do this, the machine learning lifecycle needs to be revisited and requires more research to aid practitioners [118]. Some authors point to the continuous delivery of MLOps [155], and to the end-to-end automation of the various stages of the machine learning workflow [10]. There is a need to apply software engineering principles in the ML

workflow [10], and the interaction between MLOps and existing practices within companies needs to be taken into account [155]. In addition, the opportunity to incorporate LLMs into the MLOps lifecycle has arisen because of their recent growth [121]. An appropriate data management remains a short term goal for organizations to handle [40]. In this regard, actions such as data availability [10], data standardization [40], data sharing [40], data integration [118], data collection [10], data cleaning [10], and data analysis [29] become of paramount importance.

Cloud Continuum

Data science projects promote the continuous evolution of hardware capabilities to provide the computational power and energy efficiency required [282]. For instance, the power consumption, memory, and real time constraints of microcontrollers require attention in order to deploy neural networks at the edge of the network [209]. On the other hand, FPGAs can be the cornerstone for designing the next-generation AI processors for consumer devices [282]. In fact, the use of edge and IoT devices in data science projects may lead to improvements in latency [171, 272], reliability [171], performance [7], safety [7], economy [7], privacy [189, 171, 272], energy consumption [189], and networking [189]. Yet another twist is tiny robot learning, which lies at the intersection of ML, embedded systems, and robotics, hence having to deal with their combined requirements [203]. In addition, the impact of reduced computational power can be mitigated by executing the most computationally intensive tasks on more resource-rich devices [99]. For instance, the training can be shared between both cloud and edge devices [109, 235]. In this regard, continuous learning yields the opportunity to periodically retrain the ML models based on the continuous stream of data [151], in order to raise the efficiency of the ML models [265]. In this regard, existing HPC infrastructures represent one of the most cost-effective solutions [31], and containers promote the deployment of scalable code on different operating systems and hardware architectures [32, 216].

Networking

There is a rising interest in networking among the research community in MLOps [99, 189]. For instance, the use of lightweight networks as reliability estimators may be used to predict potential task failures [169]. Some authors identify Function-as-a-Service (FaaS) technologies as enablers of MLOps patterns [266], and the deployment of already trained ML models on FaaS may unleash the use of these technologies for event-driven AI solutions [46, 296]. The continuous monitoring of ML-based applications is required for reliable performance on critical systems [180]. The deployment and redeployment of intelligent

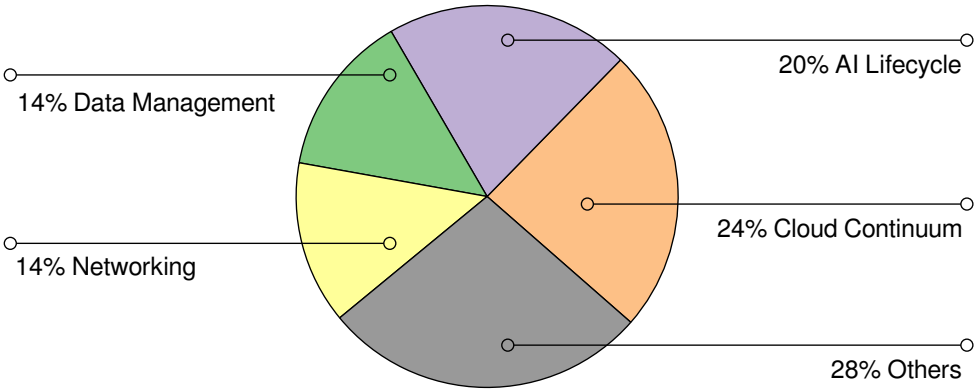

Fig. 2.7 The principal areas where opportunities and future trends for MLOps are trending.

algorithms onto heterogeneous hardware and software architectures can be alleviated by using a domain-specific language [289, 78]. Finally, traditional version control systems utilized in software engineering often lack the ability to distinguish between ML specific components such as models and datasets [137], and more research needs to be conducted in this field to better suppor the machine learning lifecycle [118].

Data science projects require the involvement of different business units, and their education and training are of paramount importance. More focus on the ML lifecycle is required, and software engineering principles such as continuous delivery need to be applied. Data management should be a pivotal point for organizations willing to benefit from AI. Being able to use hardware platforms such as FPGAs and IoT devices leads to improvements in areas such as networking and privacy. Technologies like containers and FaaS promote the deployment of scalable ML projects on different platforms and architectures. Finally, data science endeavors require additional versioning tools and frameworks for models and datasets than those of traditional software engineering.

2.4.3 What are the opportunities and future trends in AIOps? (RQ3)

This section focuses on opportunities and emerging trends in AIOps, where the inclusion of cutting-edge solutions such as DL and Transformers can be advantageous due to the complexities of these scenarios [2]. The primary topics are the AI lifecycle and its various stages, networking-related opportunities, and the various goals pursued by AIOps endeavors.

AI Lifecycle

The seamless support of the data science lifecycle is critical in ML-based solutions [223]. In [152], the authors discuss Deep Neural Network (DNN) deployment on heterogeneous hardware platforms. Multi objective optimization algorithms can be leveraged in the deployment of analytical pipelines in heterogeneous environments [78]. ML techniques can boost the efficiency of the monitoring phase and liberate IT professionals to do more innovative tasks [239]. Some authors suggest the use of generative adversarial networks for the implementation of this stage [129], others propose the automated inference of application KPIs without application-specific knowledge [110]. In addition, AIOps solutions can help monitor concept drift and suggest suitable model retraining methods [230]. A significant amount of the existing literature is focused on data management, where the continuous access to data often found in big data systems facilitates the proactive retraining of models when necessary [153]. The quality and scale of data are crucial for the observability of the system [256]. In [152], the authors identify that AIOPs can be of essential value for data lakehouses due to the complex operational challenges for SRE such as disaster recovery, backup, and restore. The interoperability between distinct components, including data collection, processing, and summarization, facilitates the use of predictive analytics over data streams [20]. The authors of [164] argue that data-driven anomaly detection is a crucial component of AIOps. In the field of root cause analysis, the use of logs to analyze certain events can become an integral part of IT operations [208]. A more ambitious endeavor is proposed in [47], where the authors address model training, packaging, and deployment. They employ AIOps to autonomously detect the state of the system, allocate resources, warn, and detect anomalies.

Networking

The complexities posed by networking technologies, such as the increasing number of devices and the growing number of services relying on connectivity, pave the way for the application of AI-based solutions to address them [197]. The myriad objectives of these network optimizations might range from minimizing energy consumption to reducing network delays [286]. Transfer learning is a promising technique for estimating the Quality of Transmission (QoT) of optical links [223]. Similarly, Deep Reinforcement Learning (DRL) optimization can raise the robustness of the network topology to tackle the inherent vulnerabilities of IoT to network failures and malicious attacks [224]. The authors in [244] use graph theory for optimizing network lifetime, which leverages the capabilities of edge devices and Software Defined Network (SDN) controllers. Similarly, deployment performance can be optimized by leveraging SDN and serverless architectures [304]. Promising results are obtained by applying genetic algorithms for minimizing network delays [80].

Objectives

The definition of certain objectives that the system will attempt to optimize is a common approach for AIOps solutions. In [34], the authors propose a system to take the burden of manual optimization of neural networks off the programmer. It outperforms existing solutions in the field in terms of performance, inference, and energy efficiency. Similarly, Hazra et al. address the difficulties posed by energy consumption and processing delays in edge environments [126]. Clustering techniques can reduce the communication load and enhance the energy efficiency of devices at the edge of the network [163]. In [80], the authors propose the use of multi objective optimization to leverage multiple converging objectives, including cost, performance, resilience, and network; subject to a set of predefined constraints. Alternatively, AIOps can serve as manner to improve software quality and engineering productivity [60, 180].

AIOps solutions can boost the various stages that comprise the data science lifecycle, including data management, monitoring, packaging, and deployment. A significant amount of the literature is devoted to data management, whereas other phases, such as monitoring and retraining of ML models, receive less attention. In this regard, the inference of metrics utilizing AI simplifies the monitoring of ML models, and the continuous stream of data facilitates their retraining. A significant portion of the literature deals with the difficulties posed by networking technologies, such as connectivity and network delays. Finally, optimizing the performance, energy consumption, and cost of existing architectures are also popular topics in academia.

2.4.4 What frameworks and architectures facilitate MLOps and AIOps? (RQ4)

This section elaborates on the various MLOps and AIOps frameworks discovered in scientific literature. A taxonomy is presented in Table 2.6, which serves to identify the differences between both methodlogies with respect to their focus areas.

Data Management

On top of the inherent difficulties of implementing MLOps and AIOps in production environments, Big Data ecosystems need to overcome the particularities associated with the volume, variety, and velocity of the data. In this regard, some authors opt to simplify the complexity of big data workloads by characterizing them into subsets, which they later train independently [153]. Similarly, the complexity of integrating big data, machine learning,

and IoT solutions has raised attention towards the training and inference orchestration of the underlying ML solutions [7] and lowered the burden for developers to work in such sophisticated environments [25]. An ML-based solution must also efficiently manage the dataset and model versions through dedicated tools and architectures [309]. The authors in [297], address the dearth of labeled datasets in ML-based solutions and propose a novel method for automating the labeling of log messages without manual labor or expert intervention. Similarly, data-driven proactive incident triage using data is utilized in [182].

Infrastructure Management

High Performance Computing represents an interesting alternative for dealing with such complex workloads, but it is challenging for data scientists due to the unique requirements and characteristics of these environments [32]. The scalability of computational resources is key for solving these large workloads. Due to this, the use of containerization solutions has yielded promising results for deploying AI on HPC systems [31]. An approach for leveraging this paradigm is the use of cloud computing resources, for which some authors highlight the importance of guaranteeing multitenancy isolation [211]. In addition, deep learning components can benefit from the deployment of training tasks over Cloud and Edge infrastructural devices [109]. Similarly, the problem of cloud resource allocation can be tackled with a hybrid multi-objective genetic algorithm [116]. In order to simplify this process, the serverless computing paradigm is an attempt to simplify the use of such resources, in which the management of the infrastructural devices is accomplished by the provider. In addition, it aspires to reduce the cost while ensuring scalable resources and inference latency [19]. To this end, a framework for the end-to-end management of the necessary resources required by ML workflows is of paramount importance [43]. The combination of serverless functions deployed on the edge of the network is gaining traction, and some authors are already implementing platforms [304] and architectures [16] to benefit from the latency improvements in comparison with the deployment of machine learning workflows on the cloud. In addition, there exist frameworks [34] specifically tailored for being able to harness the benefits of deploying artificial intelligence workloads on low consumption and limited memory edge devices. In this regard, the Internet of Things is a challenging ecosystem for developers, who often lack the expertise to work in these complex environments [25]. Due to this, there exists research on dedicated architectures [6, 72], frameworks [7, 25], and platforms [304] designed to alleviate this burden and aid in harnessing its full potential. On the other hand, well known frameworks such as Tensorflow Lite [269] and Core ML [142] are specifically tailored to utilize ML capabilities on mobile and edge devices [115]. On a similar note, yet another approach is the use of genetic algorithms for finding the configuration that

minimizes model size while maximizing accuracy on edge deployments [272]. In [308], the authors propose an architecture, coined TMLaaS for the execution of ML models on low-power IoT devices.

Lifecycle Management

The distributed training and inference of machine learning workflows originated by the emergence of these new paradigms is gaining traction in the research community [16, 66, 149]. This strengthens the necessity of managing the lifecycle and key components of AI applications through dedicated frameworks [133] and attracts research attention towards handling its various stages. In this regard, the authors in [68] propose an AI lifecycle approach for tackling the challenges of AI-based solutions from conception to production. Firstly, a semantically enhanced pipeline can automate data preparation [316]. Secondly, the training of ML workloads can be characterized into smaller pieces for distributed execution [153], and frameworks for retraining ML models to dynamically adjust to varying energy and memory constraints can speed up inference [235]. Thirdly, many authors propose frameworks and architectures that tackle the deployment stage of data science projects in different computational environments, such as the cloud [211, 109, 193] and the edge [211, 109, 16], including devices such as microcontrollers [209]. The authors of [80] propose a goal-driven framework for the operationalization of distributed analytical pipelines across the cloud continuum. In addition, the deployment of certain flavors of machine learning algorithms, such as deep learning [189] and deep neural networks [34], are also addressed. Yet another angle is to address the deployment of such workloads by maximizing resource utilization and operator revenue [59]. . Finally, some studies focus on the monitoring stage by proposing architectures [6] and frameworks [129, 133] that supervise processes and events in order to react to their deviations during runtime. The authors of [164] propose an AIOps anomaly detection framework that addresses functional and performance failures in software systems. Similarly, [230] suggests an automated pipeline for AIOps monitoring and maintenance. In summary, frameworks that are able to orchestrate ML workflows [7] and their associated resources [59] become essential to navigate through the complexity of this endeavor.

Technologies

From a technological standpoint, techniques like software-defined networks simplify network management and enable the implementation of unified services for optimal AI deployment, optimizing the available resources [304]. In [165], an architecture for the deployment of AI solutions in B5G networks is proposed. ML automation processes such as AutoML are pivotal for the maturity and efficiency of the ML models in production

			MLOps	AIOps
Infrastructure Management	HPC		[32, 31]	-
	Edge		[209, 6, 109, 235, 16, 308, 193, 72, 34, 7]	[189, 235, 304, 34, 7]
	Cloud	Generic	[109, 193, 80, 133, 153]	[80, 133, 116]
		Serverless	[16, 43]	[304, 43]
		as a Service	[149, 107, 25]	-
	Networking		[149, 165, 109, 31]	[59, 304]
Data Management	DataOps		[309]	[297, 182]
	Big Data		[25, 7]	[7]
Lifecycle Management	Generic		[7, 133, 316, 66, 68]	[7, 133]
	Monitoring		[129]	[164, 230, 182]
	Deployment		[80, 165, 32, 6, 193, 31, 34]	[80, 34]
	Training		[235]	[235]

Table 2.6 A taxonomy of the various frameworks and architectures that facilitate the adoption of MLOps and AIOps classified according to their focus areas.

environments, and more research is required [265]. Containerized solutions represent a unique opportunity for the operationalization of data science workloads since they promote the ubiquitous deployment of pipeline stages. Due to this, some architectures [72] and frameworks [31, 7] found in academia devote to this technology for the deployment and orchestration of AI workloads. Yet another technology utilized for the operationalization of predictive models is the use of API resources, as it provides a simplified abstraction on top of sophisticated prediction models, promotes resource sharing, and simplifies the consumption of the resources for the end user [107]. On the other hand, the operationalization of some flavors of algorithms captivates the attention of researchers, and the operationalization of both deep neural networks [264, 34, 209] and deep learning [66, 189, 109, 32] solutions is frequently addressed. A frequent problem is addressing the particularities of the infrastructural devices on which they are deployed.

On top of the inherent particularities of MLOps and AIOPs, data science projects frequently need to overcome the challenges of Big Data ecosystems. These platforms are often operated leveraging HPC technologies, which require the involvement of varying backgrounds of stakeholders. In this regard, containerized solutions and serverless technologies alleviate the deployment of ML-based solutions and their dependencies on different platforms and computational layers, such as the cloud and the edge. However, it also raises the manifold challenges of orchestrating distributed pipelines; hence, tools that can handle the ML lifecycle gain relevance. In this regard, the deployment of AI workloads receives the majority of the attention, while other phases like monitoring or training remain understudied. AIOps solutions are still underrepresented in the fields of infrastructure management and in the deployment of ML solutions.

2.4.5 What are the current and future fields in which MLOps and AIOps are thriving? (RQ5)

This subsection provides an analysis of the current and future fields of MLOps and AIOPs methodologies. Table 2.7 presents a taxonomy of the various fields in which MLOps and AIOps methodologies are being utilized, and highlights the areas in which each paradigm thrive.

Industry and Research

The widespread adoption of artificial intelligence solutions for increased competitiveness has reached traditional corporations. In this regard, the authors in [206] identify proactive and reactive failure management (prevention, prediction, detection, root cause analysis, remediation), and resource provisioning (consolidation, scheduling, power management, service composition, workload estimation) as the main areas in which AIOps are thriving. Some authors describe the deployment of predictive maintenance systems in stamping machines to minimize the effects and impact of unexpected failures [6]. Similarly, the redeployment of intelligent algorithms in cyber-physical production systems in Industry 4.0 remains a challenge due to the differences in reaction times, communications, and computation power in the infrastructural devices; positive feedback has been reported by experts using a domain-specific language for modeling these industrial use cases [289]. The building and construction industries have also adopted AI solutions [18], but their applications remain a challenge for large-scale projects [87]. On the other hand, innovative industries also require expertise in MLOps and AIOps should they want to incorporate the benefits of artificial intelligence into

their solutions. In the wind power industry, Wireless Sensor Networks are pivotal for the monitoring of power generation systems, but the harsh environmental conditions in which wind farms are often located make their optimal deployment troublesome [286]. In the automotive sector, the elastic deployment of training tasks over cloud and edge resources, leveraging stringent network and privacy requirements, facilitates the improvement of autonomous driving applications [109]. In space exploration, AI solutions are already applied for enhanced monitoring and diagnostics, prediction, and image analysis, but bringing AI on board remains a challenge due to the scarce computational and network resources available [99]. Next, recent advances in mobile technologies enable the development and deployment of ML-based patient monitoring right on mobile devices within the healthcare industry, but the associated challenges have not been extensively studied by the research community, and a set of recommendations is required [29]. In addition, LLMs can also be applied to healthcare by transforming data management workflows [121]. Finally, fields more traditionally associated with research are also leveraging the MLOps . In [114], facilitating the implementation of DL solutions in gravitational wave physics is discussed. On a similar note, high-energy physics requires the analysis of massive amounts of data using ML technologies and resorts to high-performance computing technologies to cope with the data storage, data transfer, and computation requirements [32].

Information Technology

One of the fields in which both methodologies are more popular is communications and networking. In particular, 5G technologies are where most of the academic attention is focused. Some authors are applying ML technologies for the rapid deployment of Quality of Transmission predictors in complex 5G network operation scenarios [223]. Next, multi-access edge computing (MEC) is a promising technology aiming to improve Quality of user Experience (QoE) of AI applications in IoT infrastructure. The authors in [304] deploy services in MEC 5G edge infrastructural devices to save costs and meet QoE requirements in massive edge data centers. Future trends for IoT involve shifting from a static to a dynamically evolving and self-organized architecture, which fits with the capabilities of 5G networks to continuously adapt and reorganize based on changing requirements. The deployment of IoT applications on these 5G edge architectures for task offloading is discussed in [163]. Supporting the specific requirements and priorities of 5G networks is cumbersome, and AI technologies can support the 5G slice deployment and orchestration for enhanced resource utilization and reduced slice request dropping probabilities [59]. In [165], the authors pursue the integration of ML techniques to optimize 5G systems. The next generation of wireless communication technologies, coined 6G, promotes the ubiquity of AI services,

		MLOps	AIOps
Research	Space	[99]	[99]
	Physics	[32, 114]	-
Industry	Factories	[6, 289]	-
	Construction	[18, 87]	[87]
	Autonomous Vehicles	[109]	-
	Health	[29, 121]	[29]
	Failure Management	-	[206]
IT	5G	[165, 223]	[59, 163, 223, 304]
	6G	[149]	-
	Networking	[29, 283]	[29, 218, 283, 286]
	Service Management	-	[2]
	Logs	-	[208, 182, 297]

Table 2.7 A taxonomy of the fields where MLOps and AIOps methodologies are thriving

and specialized frameworks for distributed AI provisioning are required [149]. Next, the appropriateness of ML technologies is discussed as a technique to perform traffic analysis and classification to identify the correct procedures and achieve the desired outcomes [218]. In this regard, AIOps is leveraged by the authors in [208, 182] for the log analysis in incident remediation, whereas [297] focuses on automated log labeling. In [2], the authors evaluate the application of AIOps technologies in IT Service Management (ITSM), highlighting the potential for predicting and resolving IT incidents in the shortest time. Finally, the many challenges associated with the heterogeneity of mobile devices have raised interest in counteracting the problems associated with the inference of DL apps [283].

Traditional corporations such as the building and construction industry and the automotive sector have already adopted AI-based solutions, but more effort is required for their implementation in larger endeavors. Similarly, more innovative sectors, such as the wind power industry or even space exploration, are investing in AI solutions to address the challenges associated with the harsh environmental conditions in which they operate. Academic disciplines, such as physics, are devoted to MLOps for coping with stringent data requirements. However, the area in which MLOps and AIOps are more prevalent is communications and networking. The rise of 5G and 6G architectures has yielded some challenges and opportunities, such as slice deployment and traffic analysis, for which AI-based solutions are ideal. It is anticipated that much of the current 5G effort will gradually shift toward 6G technologies. Service management and log analysis are highly attractive areas for AIOps solutions, whereas physics, autonomous vehicles, and traditional factories are more akin to MLOps.

2.5 Related Work

We have compiled all pertinent studies and reviews in the fields of MLOps and AIOps to provide the rationale for this work and situate it within the body of existing research. There is one study offering a general perspective on AIOPs, three studies on specific fields of AIOPs, and one on MLOps. To the best of our knowledge, there has not been a study that conveys both MLOps and AIOPs and provides as detailed a broad view of both fields as this one.

In [239], the authors perform a multivocal literature review (MLR) in which they identify that the adoption of AIOps helps in monitoring IT work and improves human-AI collaboration. There are, however, concerns about the effectiveness of AI and ML and the quality of the quality of the data utilized to obtain the results. Notaro et al. [207] provide five different categories and fourteen subcategories for the categorization of failure management IT solutions using AIOps. In [316], the authors review and categorize existing works around three key processes in log processing, such as log enhancement, log parsing, and log analysis. They finalize future directions and development trends in the field of log research. Lima et al. [174] provide a systematic literature review on practices, maturity models, roles, tools, and challenges for MLOps and establish that this methodology is still in its infancy, leaving room for future academic studies that will guide organizations. A systematic literature review of existing work and challenges in the field of IT risk prediction is offered in [2], which reveals ML classifiers as the preferred method for implementing IT service management and highlights the importance of adopting more advanced state-of-the-art methods, such as DL and Transformers. In [166] the authors focus on the definition and architecture of MLOps and perform a literature review (up to May 2021), a tool review, and interviews with experts in the field. They conclude that model development and benchmarking have received more attention in the academic community than operating ML systems, which is still challenging today. The authors of [48] conducted a survey regarding the opportunities and challenges of AIOps. They suggest that, with the growth of IT infrastructure, AIOps is the only promising solution to cope with it. Numerous techniques, such as anomaly detection, root-cause analysis, failure predictions, automated actions, and resource management, are utilized in AIOps. They have found that many of the solutions focus on detection and root cause analysis, but automation is still limited.

Table 2.8 provides a comparison framework for this SLR with each of the studies described above. For the purpose of evaluating our work, we have established four different criteria. Firstly, we have followed the PRISMA methodology for systematic literature reviews to transparently report the reasoning for doing the review, what has been accomplished, and the discoveries of the review. Secondly, this manuscript is the most contemporary, as it contains studies up to 2023. Thirdly, it is the only one that offers a joint perspective on both

Paper	Type	Time frames	Topic	Focus
This study	SLR [1]	2018 – 2023	MLOps, AIOps	Challenges, Architectures, Future Fields
Rijal et al. [239]	MLR [2]	2017 – 2021	AIOps	Benefits, challenges, and future.
Notaro et al. [207]	SMS [3]	1990 – 2020	AIOps	Failure Management
Zhaoxue et al. [314]	Survey	up to 2020 [4]	AIOps	Logs
Lima et al. [174]	SLR [1]	up to 2021	MLOps	Maturity Models, tools and challenges.
Ahmed et al. [2]	SLR [1]	2000 – 2022	AIOps	Risk Prediction
Kreuzberger et al. [166]	MM [5]	up to 2021	MLOps	Definition and Architecture
Cheng et al. [48]	Survey	up to 2023	AIOps	Opportunities and challenges

[1] Systematic Literature Review [2] Multivocal Literature Review [3] Systematic Mapping Study
[4] Time frame not explicitly stated [5] Mixed Method

Table 2.8 Related works conducted in areas covered by this research.

MLOps and AIOps methodologies, as we think they have a strong influence on one another. Finally, the focus of this study is the widest, as it comprises not only a general overview of the challenges and benefits of these methodologies but also a deep dive into their current and future use in both industry and academia.

2.6 Conclusions and Future Work

The overarching goal of this SLR is to offer insights into the adoption of MLOps and AIOps methodologies in both industry and academia. We have conducted an in-depth search of the scientific literature based on the search terms described in Figure 2.1, and we elaborate the discussion on the research questions outlined in Table 2.2. The necessity for this SLR stems from the fact that the implementation of data science projects in production environments is a challenging endeavor in which MLOps and AIOps can play a facilitating role. However, organizations and stakeholders need to have both a collaborative culture and a unique cross-domain skillset of software engineering, data science, and IT operations. Data management plays a significant role in these projects and must be planned accordingly. In this regard, the complexity is amplified by big data ecosystems and their unique requirements and characteristics. In addition, recent computing paradigms such as cloud and edge computing require innovative solutions due to the distributed and heterogeneous nature of the infrastructural devices they comprise. To circumvent these challenges, more focus is required on applying software engineering principles to the ML lifecycle. In this regard, technologies such as containerization, data, and model versioning, FaaS, and serverless are the cornerstones for supporting this lifecycle. The monitoring stage, which allows the

retraining and redeployment of the relevant components of the architecture subject to the various drifts of the production environments, can benefit from AIOps solutions for relevant KPI inference. On the other hand, AIOps solutions are often focused on solving the myriad networking requirements, and hardware configurations of modern ecosystems. Due to this, the use of data science orchestration frameworks can be beneficial since they are specifically tailored to address various stages of the ML lifecycle (e.g., re-training, re-deployment, monitoring, versioning). The reviewed manuscripts showcase that AI-based solutions are no longer restricted to academia but have reached not only innovative industries such as space exploration and the wind power industry, but also traditional corporations such as the construction and automotive sectors. Finally, the rise of 5G and 6G technologies and architectures leveraging MLOPs and AIOps methodologies pose an opportunity for AI-based solutions. As for future work, since AIOps is such a new trend, this manuscript provides a shared perspective on the challenges associated with the adoption of both MLOps and AIOPs. We reckon sufficient AIOps-related manuscripts will emerge in the near future to be able to distinguish the challenges from those of MLOps. In addition, the recent upsurge in LLMs will spread to MLOps and AIOs, and further analysis on this subject will be necessary.

We can only see a short distance ahead, but we can see plenty there that needs to be done.

Alan Turing

3

Modeling Advanced Analytical Services

The work presented in this chapter has resulted on various scientific publications. The initial germ was presented in the 5th International Conference on Smart and Sustainable Technologies [89]. An extension of that manuscript was published on 2020 in Sensors journal [79], which received an impact factor of 3.576 that year and was ranked as a Q2 in JCR. This chapter delivers SC3, TC2, TC3; therefore fulfilling O2.

In the smart city context, Big Data analytics plays an important role in processing the data collected through IoT devices. The analysis of the information gathered by sensors favors the generation of specific services and systems that not only improve the quality of life of the citizens, but also optimize the city resources. However, the difficulties of implementing this entire process in real scenarios are manifold, including the huge amount and heterogeneity of the devices, their geographical distribution, and the complexity of the necessary IT infrastructures. For this reason, the main contribution of this paper is the PADL description language, which has been specifically tailored to assist in the definition and operationalization phases of the machine learning life cycle. It provides annotations that serve as an abstraction layer from the underlying infrastructure and technologies, hence facilitating

the work of data scientists and engineers. Due to its proficiency in the operationalization of distributed pipelines over edge, fog, and cloud layers, it is particularly useful in the complex and heterogeneous environments of smart cities. For this purpose, PADL contains functionalities for the specification of monitoring, notifications, and actuation capabilities. In addition, we provide tools that facilitate its adoption in production environments. Finally, we showcase the usefulness of the language by showing the definition of PADL-compliant analytical pipelines over two uses cases in a smart city context (flood control and waste management), demonstrating that its adoption is simple and beneficial for the definition of information and process flows in such environments.

3.1 Introduction

In recent years, the concept of smart cities has emerged in response to the challenges posed by the continuous development of urban infrastructures and the increase in population density. The main objective of this paradigm is to enhance the management of the city by providing smarter, safer, and more sustainable ecosystems [4]. In order to properly address these challenges, specific services and systems have to be developed and provided to citizens. This leads to an improvement of their quality of life; on the other hand, more complex ICT infrastructures become necessary.

In this context, Artificial Intelligence (AI) is a key enabling technology, since it provides the necessary foundations for the intelligence and resilience of future cities. Furthermore, ICT tools can deal with the diverse application domains that exist in a city. For instance, Lim et al. [173] conducted an intensive study of the related literature to identify twelve distinct domain categories: “smart device”, “smart environment”, “smart home”, “smart energy”, “smart building”, “smart transportation”, “smart logistics”, “smart farming”, “smart security”, “smart health”, “smart hospitality”, and “smart education”. These categories share the objective of building an integrated and advanced intelligent information ecosystem to enable a framework that will boost the socioeconomic growth of the city. However, extracting valuable information from the collected data produced requires effective techniques, tools, and software technologies to collect, store, analyze, and visualize large amounts of data from the city environment, its organizations, departments, agencies, and citizens.

Some authors [161] have stated that the ideal model of a smart city is mainly based on the following pillars, in which AI plays a key role:

1. Smart sensors enable the collection of the necessary data and transform the city into a smart city. They maintain the city connected and the stakeholders informed, and without them, the rest of subsystems would not be able to function correctly.

2. Smart citizens are undoubtedly the fundamental part of a smart city since their active involvement makes it possible to perform these initiatives.

3. Smart services enable citizens and administrative entities to actively participate in the control of the city. They are based on information technologies that help to control the different subsystems that comprise the smart city.

In isolation, these statements present challenges that have to be addressed. Nevertheless, the bottom line is offering a unified solution to enable a coordinated and effective flow of information (both from sensors and citizens) and the execution of processes among the potential advanced services offered by the city. Next, the different pillars and the problems detected by each one are introduced in detail.

The growing adoption of sensors being deployed in cities, smart sensors, has been significantly favored by certain paradigms such as the Internet of Things (IoT). This paradigm expands the definition of smart cities, which become more connected. However, this innovation will largely depend on the capabilities of the underlying technologies to leverage the information and resources supplied by the vast volume of emerging devices. Therefore, the key to achieve intelligent cities depends on the adequate analysis of the connected data sources. In addition, IoT devices offer the possibility of lowering the computations down to the edge layer, that is to say, closer to where data are generated and collected. This can be particularly attractive in contexts where the latency caused by the interactions among the elements of the IoT environment and the cloud is too high [222] or crucial. These two technological advances, a greater distribution of knowledge and its computation and the appearance of more advanced sensors capable of executing certain computations reducing the latency, must be included in the services proposed by cities. This is because in spite of presenting a great technological complexity, they can significantly improve the final experience of citizens.

Currently, knowledge, computing, and storage infrastructures tend to be distributed. This implies that systems, architectures, algorithms, and techniques designed for cities have to face additional requirements when being implemented (e.g., scalability, partitioning, and distribution).

The smart city concept lies at the intersection of city administration, citizen value creation, local business, ICT development and application, urban Big Data, economics, and sociology, citizen engagement (smart citizens) being a key element of most definitions of smart cities. Additionally, Information and Communication Technologies (ICTs) play a crucial role as the drivers and enablers of citizen participation, being the necessary technologies for processing the information with AI and providing in this way a better service to the relevant actors. In this context, it is essential that new services have functionalities like self-monitoring in

order to guarantee their stability and correct maintenance. On the other hand, users need to interact appropriately with the smart city ecosystem. Therefore, it is necessary to provide services with the possibility of sending notifications, alerts, and mechanisms for actuating on the devices.

In recent times, the continuous advances in AI have revolutionized intelligent solutions in a plethora of domains. Specifically in the smart city domain, it becomes clear that AI must be the cornerstone to generate new valuable services, the formerly called smart services. The main areas of improvement in which AI affects cities are efficient resource management [202] and the improvement of the citizens' quality of life [255]. Regarding the data being generated by smart cities, two main considerations must be taken into account: its massive volume, due to the growing generation of data by citizens and devices, and its sparseness, due to the ubiquitous deployment of various types of sensors. These characteristics require a new computing paradigm to offer location-aware and latency-sensitive monitoring and intelligent control [267].

As a result of this new architectural landscape, new AI approaches have arisen in recent times, with a manifold of implications in terms of operationalization: (i) federated learning is a distributed machine learning paradigm that enables privacy-aware learning over distributed data by using edge computing and (ii) stream learning techniques on the edge, in which nearly real-time learning algorithms can be deployed closer to IoT devices for lower latencies, resiliency, efficient use of bandwidth, and compliance. This work indeed addresses this particular strand of challenges, which is collectively referred to as "operationalization" since the proposed language does not only focus on the definition, but also includes functionalities for the deployment and monitoring of analytical services.

However, the scientific community working in the fields of AI and Data Science (DS) has mainly been focused on the development of new algorithms and learning techniques, in many cases under laboratory or experimental setups. Consequently, transferring these prototypes and tests to real production environments, as is very often demanded by smart city stakeholders, has proven to be difficult to achieve with those approaches. This article stresses this vision, namely the way in which smart data-based services are provisioned in real infrastructures.

In this work, a proposal to tackle this is provided from the ICT perspective: a description language that eases the definition and operationalization of these data flows and the associated processes. By virtue of this language, the difficulties emerging when working with complex disciplines such as Big Data, AI, or IoT, alongside those explained above: (i) infrastructure distribution, (ii) AI operationalization, and (iii) the large number and ubiquity of sensors, can be minimized. Summarizing, this manuscript revolves around the different needs presented by the services and use cases that fall within the smart city domain, such as: (a) the definition

of complex and composite analytical services through different phases in which various actors and data sources can be involved; (b) the deployment of these services in the form of analytical pipelines on very diverse ICT infrastructures (i.e., edge, fog, and cloud); and (c) the provision of heterogeneous actuation mechanisms, using the devices and sensors (actors), in the form of alerts or actuators. As a consequence, the main contribution of this article is a domain specific language coined as the Analytical Pipeline Definition and Deployment Language (PADL), which is devoted to supporting the operationalization of analytical pipelines in heterogeneous infrastructures by enabling the definition and provision of complex analytical services in the domain of smart cities. In addition, data scientists do not need to fathom the infrastructure and underlying technologies in detail; instead, they can use annotations so the models are operationalized appropriately, in a reproducible manner. Finally, certain tools for the smooth deployment of analytical pipelines in production environments are presented.

The proposed language is validated using two use cases in the domains of flood control and waste management. In both cases, we exemplify how to use PADL to define the pipeline and the technical complexities in detail, as well as the benefits it brings to the city ecosystem (ICT infrastructure and actors). To the best of our knowledge, there are no languages or tools similar to the one proposed in this article. Therefore, instead of comparing its efficiency against other works, the demonstrations focus on highlighting its usefulness.

The rest of the paper is organized as follows. Section 3.2 presents related work in the scope of provisioning analytic services in smart cities and technologies related to the deployment in the production of analytical processes. Section 3.3 describes the main features of the proposed language PADL. Section 3.4 elaborates on the technical details of the language implementation, along with a set of auxiliary tools that assist in its use. Section 3.5 presents the two use cases, which serve as the validation of the practical utility of the PADL language. Finally, conclusions and future work are presented in Section 3.6. A preliminary study, which did not include the implementation and tools provided in this paper and was not specifically focused on the smart city domain, was presented as a conference paper [89].

3.2 Deploying Data Analytics in Smart Cities

Different software solutions have been proposed in the literature to provide smarter, safer, and more sustainable cities through new paradigms like Big Data, AI, or IoT. The main objective of these solutions is to generate innovative services to citizens leveraging AI techniques. This section is divided into two parts: the first introduces different research projects and initiatives that deal with advanced analytical services that can be integrated into the city

ecosystem; the second part explains the technological background necessary to understand the difficulties of deploying these solutions and services in production environments.

3.2.1 Analytic Services for Smart Cities

There has been an enormous increase in the data generated in urban ecosystems. These data can be useful for providing high-value services in different domains by using analytical techniques. Arasteh et al. [11] provided a review of the concept of smart cities, its motivations, and applications. Additionally, they examined IoT technologies for smart cities and their main components and features. Sanchez et al. [246] performed a study on the future technological challenges of smart cities, where they pointed out the complexity of the analytical process flows needed in this context. For instance, in the smart city domain, the work in [162] proposed a Big Data-driven analysis and a cloud-based analytic service that utilizes urban environment indicators such as quality of life. Hossain et al. in [130] analyzed GIS data, historical data, and other parameters such as building slope, flow accumulation, land use, soil types, and distance from the river in order to explore individual residential and business buildings for flooding risk in Birmingham (U.K.). Mazhar et al. [238] proposed a system with various types of sensors ranging from smart homes, vehicular networking, and weather, water, and smart parking sensors, surveillance objects and a four-tier architecture devoted to manage IoT sources, communications, and data, using Big Data technologies for processing and serving data. This way, they analyzed both in batch and real-time smart homes, smart parking, weather, pollution, and vehicle datasets aiming at optimizing the urban planning and enhancing the future city development. These types of approaches could be potentially simplified by using the PADL syntax, since it provides an abstraction layer over the underlying infrastructure deployment and eases the analytical pipeline conceptualization.

Citizens are one of the pillars of smart cities, and the article presented by Aguilera et al. [1] focused on how to accelerate the generation of citizen-centric apps that exploit urban data in different domains. Furthermore, there are very interesting studies that try to take advantage of citizen potential (data and participation) through applications, based on IoT or AI, and by means of initiatives like citizen science. The study proposed by Lopez-Novoa et al. [177] is a clear example, explaining how to use an IoT communications platform to promote a citizen science project, where individuals collect, categorize, and sometimes analyze scientific data. Furthermore, there are studies like [85] that acknowledged the huge amount of information citizens are exposed to and proposed a cognitive-driven, personalized information system to minimize potential cognitive overload issues.

In addition to the previous works that offered overhauls on the benefits of analytical data in the context of smart cities, there are currently many works focused on solving very specific

problems or use cases in the smart city domain, like that presented by Cerchecci et al. [45] for optimizing waste collection or that presented by Obinikpo et al. [210] for smarter health care in smart cities. Additional literature in the field of smart city analytics was explored by Wang et al. in [292], where a survey of deep learning techniques in smart cities was offered. Concretely, it examined algorithms applied to video analytics of smart cities in terms of different research categories: object detection, object tracking, face recognition, image classification, and scene labeling. Moreover, Saggi et al. in [245] deeply explored the concept of Big Data analytics, and a methodical analysis for the usage of Big Data analytics in various domains including smart cities was presented. Lavalle et al. in [168] proposed a methodology based on visualization techniques, which assist users, the goal being to improve the evidence-gathering process, in this way contributing to the optimization of the city resources.

Urban mobility is yet another field of study in smart cities. Vergis et al. in [287] proposed a low cost traffic monitoring system using IoT devices and fog computing and provided a system capable of estimating traffic flow, which can be exploited by the authorities. Tekouabou et al. [268] proposed a system that integrates IoT and set-based regression models to predict the availability of free places for parking, hence reducing urban congestion. Hîrţan et al. in [128] presented a reputation system with the purpose of providing users with an optimal travel route, focusing at the same time on privacy and confidentiality.

However, all these works focus on building dedicated systems over a specific infrastructure. The approach presented in this article is positioned at a higher level of abstraction from the ICT perspective. The main idea is to provide an analytical service definition and deployment language in the domain of smart cities. This way, these definitions can be integrated with the existing city ecosystem/model and ICT infrastructure. Additionally, each new advanced analytical service does not have to be conceptualized as a particular solution or independent system, but be defined appropriately according to the proposed language aiming at being deployed in the existing infrastructure.

3.2.2 Technological Background

There are two main approaches in the context of the deployment of analytical pipelines in production environments:

- Deployment frameworks: In this field, there are novel works such as MLflow [306], which provides tools for packaging and deploying analytical pipelines in different environments. ML.NET [170] is another open-source framework proposed by Microsoft to deploy machine learning pipelines. For typical operational workflow operationalization, on the other hand, Verma et al. [288] proposed a cluster manager to execute

hundreds of workloads across thousands of machines. This project was the cornerstone for Kubernetes [95]. Another effort towards the orchestration of workloads in distributed environments is Docker Swarm [134], which offers less features than Kubernetes, but at a lower technological footprint. However, both Kubernetes and Docker Swarm are general-purpose deployment frameworks and do not focus specifically on ML or AI.

- Description languages: In the machine learning domain, the Predictive Model Markup Language (PMML) [111] was one of the first solutions that tackled the problems associated with the operationalization of AI and ML models. Subsequently, Portable Format for Analytics (PFA) [227] was conceptualized claiming that, in contrast to PMML, it is an extensible language for the definition of pre-processing and post-processing code, provides better features to create analytical workflows, integrates easily with distributed and event-based data processing platforms, and is safer to use within IT operational environments.

PADL is conceived of to coexist at the intersection of these two approaches, with a twofold goal: first, to be specific to the analytical scope and to understand its nuances; and secondly, to be sufficiently expressive, providing a means for the deployment of these pipelines in heterogeneous environments, technologies, and infrastructures. Table 3.1 offers a comparison of the aforementioned technologies in conjunction with some trends found in academia, and all of them are evaluated against the criteria explained below:

1. Deployment awareness, allowing for the definition of the restrictions an analytical pipeline needs to adhere to when deployed in production.
2. Domain orientation, providing annotations specific for the analytical domain.
3. Interacts with technologies already existing in the infrastructure (i.e., Apache Spark [307], Apache Flink [42]).
4. Permits the definition of the entire pipeline alongside the annotations in a single text file, so it can be versioned and integrated into continuous integration and delivery workflows.
5. Has a low technological footprint, facilitating the use of existing infrastructures with heterogeneous devices.
6. Enables the deployment of analytical pipelines in different layers of the architecture (i.e., edge, fog, cloud).

	Deployment Awareness	Analytics Oriented	Technology Agnostic	Text Based	Small Technological Footprint	Multilayer Awareness
PFA	✗	✓	✗	✓	✗	✗
PMML	✗	✓	✗	✓	✗	✗
MLflow	≈	✓	≈	✗	✓	✗
ML.NET	≈	✓	✗	✗	✗	✗
Kubernetes	✓	✗	✗	✓	✗	✗
Docker Swarm	✓	✗	≈	✓	≈	✗
Stratum	≈	✓	≈	✗	≈	✗
DEEP-Hybrid-DataCloud	≈	✓	≈	✗	≈	✗
PADL	✓	✓	✓	✓	✓	✓

✓: full support, ≈: partial support, ✗: not supported

Table 3.1 Technologies for deploying analytical pipelines in production environments.

Both PMML and PFA excel at the formalization of analytical pipelines, but fail in terms of integration with existing technologies. Additionally, they do not contemplate the deployment over different computation layers, nor do they consider distributed processing. On the other hand, Kubernetes and Docker Swarm are proficient at deploying ecosystems at multiple nodes, but are general-purpose tools; hence, they lack the ability to understand analytical pipelines towards distributing them properly. Finally, MLflow and ML.NET shine in training and packaging analytical pipelines, but only cover the deployment to some extent, without being able to fully utilize the existing technologies, nor being aware of the various computational layers (e.g., edge, fog, cloud). Among the most recent works, we do not find similar approaches, and we can only review the Stratum and DEEP-Hybrid-DataCloud frameworks. Stratum [25] focuses on analytics, but it is not text based nor does it focus on multilayered deployments. The DEEP-Hybrid-DataCloud [101] framework is very similar to Stratum, and it lacks the awareness for being deployed in multiple processing layers. Summarizing, currently, there is no tool that can efficiently compete in all the different aspects of the operationalization of data-based pipelines. Those of the analytical domain do not easily adopt the benefits of distributed environments. Conversely, deployment tools lack the capacity of understanding analytical pipelines and cannot parallelize them effectively across edge and fog environments.

3.3 PADL Specification

Our proposed PADL domain specific language enables the definition of analytical pipelines and provides a specific syntax for enriching it with functional and non-functional requirements. Among other features, PADL: (i) abstracts the user from the underlying infrastructure and technologies; (ii) enables the deployment of the analytical pipeline in

various computational layers (edge, fog, cloud); (iii) promotes the operationalization of these pipelines over continuous integration and deployment environments; (iv) provides a means for specifying performance, security, and monitoring capabilities alongside its definition; and (v) simplifies the development process, since data scientists can test the models in different infrastructures before deploying them in production. PADL is a fresh twist on the everything-as-code [310] trend and can be written/edited with a simple text editor in YAML [271] format, which makes it readable by humans and machines. Documents written in PADL should be validated against a schema that can be found in the official PADL repository [77].

3.3.1 Ecosystem

An architecture that reconciles various computing layers within a Big Data paradigm is proposed by Díaz-de-Arcaya et al. in [88]. This architecture expands across the edge, fog, and cloud computing layers and is oriented toward the deployment of analytical pipelines in heterogeneous infrastructures. Among the assorted components that integrate this architecture, the focus of this research work is to provide a basic building block for the implementation of the Analytic Orchestrator and Life Cycle Management module, a domain specific language called PADL. This component is able to understand a PADL document and deploy the defined analytical pipeline in the infrastructure. PADL, on the other hand, serves as the definition for the analytical pipeline and offers annotations to abstract the user from the shortcomings of production deployments (e.g., infrastructure details, network constraints, technologies).

The PADL deployment flow is schematically depicted in Figure 3.1. It comprises three stages: (i) training and packaging, (ii) pipeline orchestration, and iii) deployment and monitoring. Initially, the data scientist builds his/her analytical pipeline using his/her preferred library and packages the different models either using a definition language (e.g., PFA [227], PMML [111]) or a machine learning specific packaging system (e.g., MLflow Projects (https://mlflow.org/docs/latest/projects.html)). Then, he/she creates a PADL document with the definition of the analytical pipeline and enriches it with deployment specific annotations (i.e., how the pipeline should be deployed in the production environment) and machine learning related annotations (i.e., which action should be taken if the performance of a model drops below a certain threshold). The data scientist sends (1) the model over to the architecture, where the (a) PipeHub module persists (2) both the trained models and the PADL document in a (b) database. In addition to this, it sends (3) a message to the (c) Orchestrator through the (d) Message Bus, which is the module in charge of the communications across the whole architecture, and with the infrastructure (7). Then, the Orchestrator retrieves (4) a document from the database with information related to the infrastructure in which the

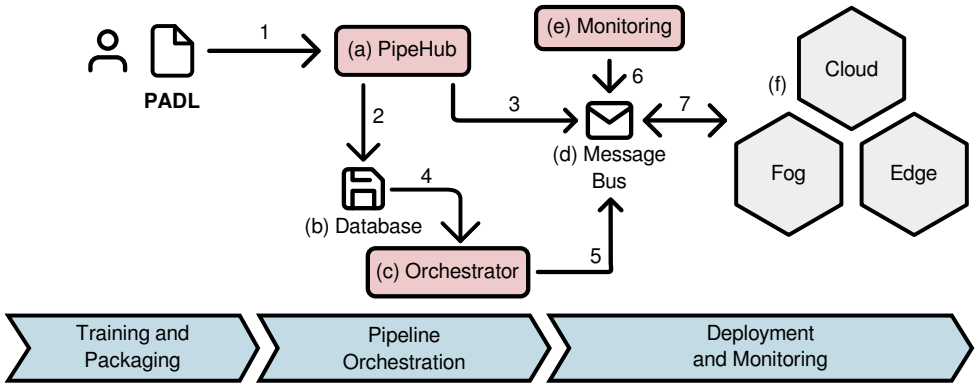

Fig. 3.1 Reference PADL architecture.

pipeline will be deployed. At this point, the Orchestrator has both the PADL and the infrastructure documents and is capable of making a decision on where each stage of the pipeline will be deployed in the production environment. This information will be communicated (5) to the infrastructure through the (d) Message Bus . Finally, the (e) Monitoring module ensures (6) that the pipeline is deployed and operates according to the annotations. Should an anomaly occur, a notification is sent to the stakeholders through a predefined channel.

The devices that conform the infrastructure use an agent for communicating with the architecture. This agent publishes the capacities of the node, so that a map of the whole infrastructure can be built. The communication between the Orchestrator and the infrastructure is accomplished through a publish-subscribe protocol (e.g., MQTT, Kafka, CoAP [28]). This technology has been chosen because, especially in the lower layers (i.e., edge), a single model may be deployed in thousands of devices; relying on having to send the same message across the network to that many devices would not be feasible. On the contrary, a publish-subscribe protocol would only require the message to be published in a single topic to be spread across the network.

In a smart city context, the PADL language can offer additional value since analytical pipelines can rapidly be defined and deployed without having to modify the existing infrastructure. This benefit enables modern cities to easily provide new services to citizens and to be able to adapt to new contexts in an agile way. In addition, it is worth mentioning that due to the monitorization, notification, and actuation capabilities supported by PADL, as soon as a specific event in the city is identified, a notification mechanism can be triggered and inform the relevant stakeholders. In addition, specific actuators can be managed by PADL in order to address the problem.

3.3.2 Language Details and Application Preconditions

The Analytical Pipeline Definition and Deployment Language (PADL) is a domain specific language for the description of analytical pipelines, including a definition level intended for their operationalization that covers the deployment characteristics necessary to implement them on previously defined and available infrastructures. A data analytic pipeline is comprised of different data analysis phases, materialized in a compound of command-line tools and custom scripts that implement each of these analytic processes. For instance, an analytical pipeline could be used to detect traffic jams and react accordingly through suitable mobility plans. In this context, PADL would act as the framework for defining all the necessary analytical stages, from the processing of vehicle GPS signals, to decision-making in the form of alternative routing to be used by citizens. Furthermore, it allows the automation of all the processes involved in existing heterogeneous IT infrastructures.

PADL is a description language that assists in the deployment of analytical pipelines considering very heterogeneous production environments. It provides a means for the definition of domain agnostic analytical pipelines, as well as for the specification of the automation and operationalization criteria of the implementation processes. Figure 3.2 showcases the definition process of analytical pipelines via PADL and how this language enables the enrichment of the analytical models and data processing stages, by utilizing the available features for the operationalization. At first, the data scientist trains his/her models and packages them in a certain format. Afterwards, PADL queues promote the definition of a flow with these models. Then, models can be be deployed in different layers (e.g., edge, cloud) and be constrained to execute under different infrastructure conditions (e.g., two CPU cores, 8GB RAM), all this with no knowledge of the infrastructure by the data scientist. Finally, watches enable the definition of actuators and notifications depending on the desired monitoring conditions. PADL is intended to be a simple, normalized, and interoperable language; therefore, it has been built as a subset of the YAML serialization language, with a relatively simple syntax and designed to be comprehensible. The pipeline stages and their characteristics can be adequately represented as combinations of lists, mappings, and scalar data (simple values). This way, any library or analytical platform willing to utilize PADL will need to import and export its definitions in this simple format.

The simplest example of PADL is that composed of an empty pipeline, as can be seen in Listing 3.3.

In Section 3.3.3, an extended syntax explanation is provided, in which each of the elements that comprises a pipeline is explained in further detail.

Finally, it is important to establish the application conditions to be met and the main objectives to be achieved through the use of PADL. The application conditions are only

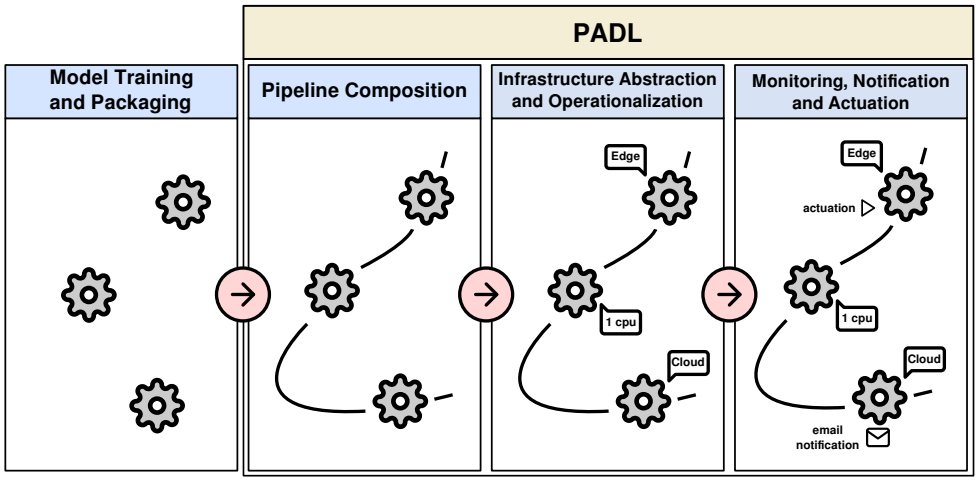

Fig. 3.2 Definition and operation of analytical pipelines through the functionalities provided by PADL.

```yaml
version: 0
pipeline:
  small:
    model: null
queues:
  input: null
  output: null
```

Fig. 3.3 Minimum example of a PADL document.

two: (i) the models of each stage of the analytical pipeline must have been previously selected, trained, and packaged in order to be specified in PADL; (ii) the provisioning and configuration of the infrastructure are not the responsibility of PADL; it will only be in charge of defining the restrictions and characteristics that should be met for the operationalization to be successful. The main PADL objectives orbit around the following points:

1. It should aspire to become a standard for the definition of analytical process flows.
2. It should help data analysts and domain experts overcome the technological barriers that prevent greater success in the operationalization of analytical processes.
3. It must be expressive enough to allow the definition of analytical pipelines as a chain of processing stages, in which each stage represents a model and its characteristics.

4. It must be able to be easily integrated into different analytical platforms and systems responsible for the operation of large analytical workloads.

3.3.3 Language Syntax

In this subsection, a simple PADL document is used to introduce the language. Then, the complete specification is discussed in further detail. The PADL document portrayed in Figure 3.4 specifies an analytical pipeline composed of a single model that has been packaged using the Portable Format for Analytics language.

The pipeline in Figure 3.4 will deploy the model *decissionTree.pfa* into the infrastructure. As there are no other attributes defined for this particular pipeline, we will assume the existing elements already fulfill the requirements for its execution. This model will monitor a text file, in this case the logs generated by an existing web server, and will populate another text file with the results.

```
version: '1.0'
pipeline:
  decissionTree:
    model: decissionTree.pfa
queues:
  input: apache2-log
  output: padl-log
```

Fig. 3.4 Simple example of a PADL document.

Even though this is a very simple example, it introduces the use of queues as it can be seen in Figure 3.5. The purpose of this code structure is to specify the inputs and outputs for the different models. For the sake of simplicity, reading and writing to a text file is herein considered, yet a myriad of other different possibilities exist.

The complete PADL schema can be observed in Figure 3.6, in which pipeline and queue structures were discussed previously. These are the most important entities within a PADL document, which should contain both. The former defines which data transformation processes or models need to be run as part of the pipeline, whereas the latter defines how these models communicate with their environment and among themselves. Another important entity is constraints which can be seen in Figure 3.7. This structure may have two children: (i) the **node** attribute specifies variables such as the operating system, the hostname, or the layer in which the model must run (e.g., this model must only be run in the edge layer); and (ii) the **model** itself, which defines constraints specific to the machine learning domain, such

```
queues:
  apache2-log:
    format: file
    path: 'file://var/log/httpd.log'
  padl-log:
    format: file
    path: 'file://var/log/padl.log'
```

Fig. 3.5 Example of the queues keyword.

as **max_execution_time**, which defines the maximum time granted for a model to produce its output (e.g., a predicted value of the target variable in predictive modeling), beyond which it will halt its execution.

In Figure 3.8 the **deploy** structure determines the behavior during an update (i.e., in case the efficiency of the deployed model has decreased and needs to be redeployed) and during a rollback (i.e., the previous deployment is not performing as expected, so that a previous version will be deployed). Both cases support the **on_failure** attribute. In the case of a rollback failure, the counteracting behavior can be to declare the model as deprecated and stop trying to redeploy it, or to keep trying until it succeeds. However, an update supports both **retry** and **exit** attributes in the case of failure, but it also supports the **rollback** of the model to the previous version.

Another children of the model structure is the **environment** attribute. This differs from the previous ones in which is not constrained to a predefined set of attributes, but it can hold any key-value pair. It is commonly used for specifying variables the model will have to use in order to meet its purpose (e.g., specifying the full path to the version of Java the model needs). In addition to this, instead of having to write all the variables one at a time, it accepts a file path, so all variables can be added to that file and shared across the whole pipeline.

Labels is another keyword that behaves similarly to the **environment** keyword, in which it can accept any amount of non-predefined key-value pairs. A model annotated with a label will match the devices in the infrastructure annotated with the same label. This is useful to deploy a model in a geographical location (e.g., legislation differs from European to American servers) or to deploy a model in a device subject to high security policies. An example of it can be seen in figure 3.9.

The **resources** attribute, which can be seen in Figure 3.10 is used to specify the system requirements (e.g., CPU, memory) reserved by a certain model to fulfill its task. The *Orchestrator* depicted in Figure 3.1 can use this information to guarantee no device is planned over its resource limits.

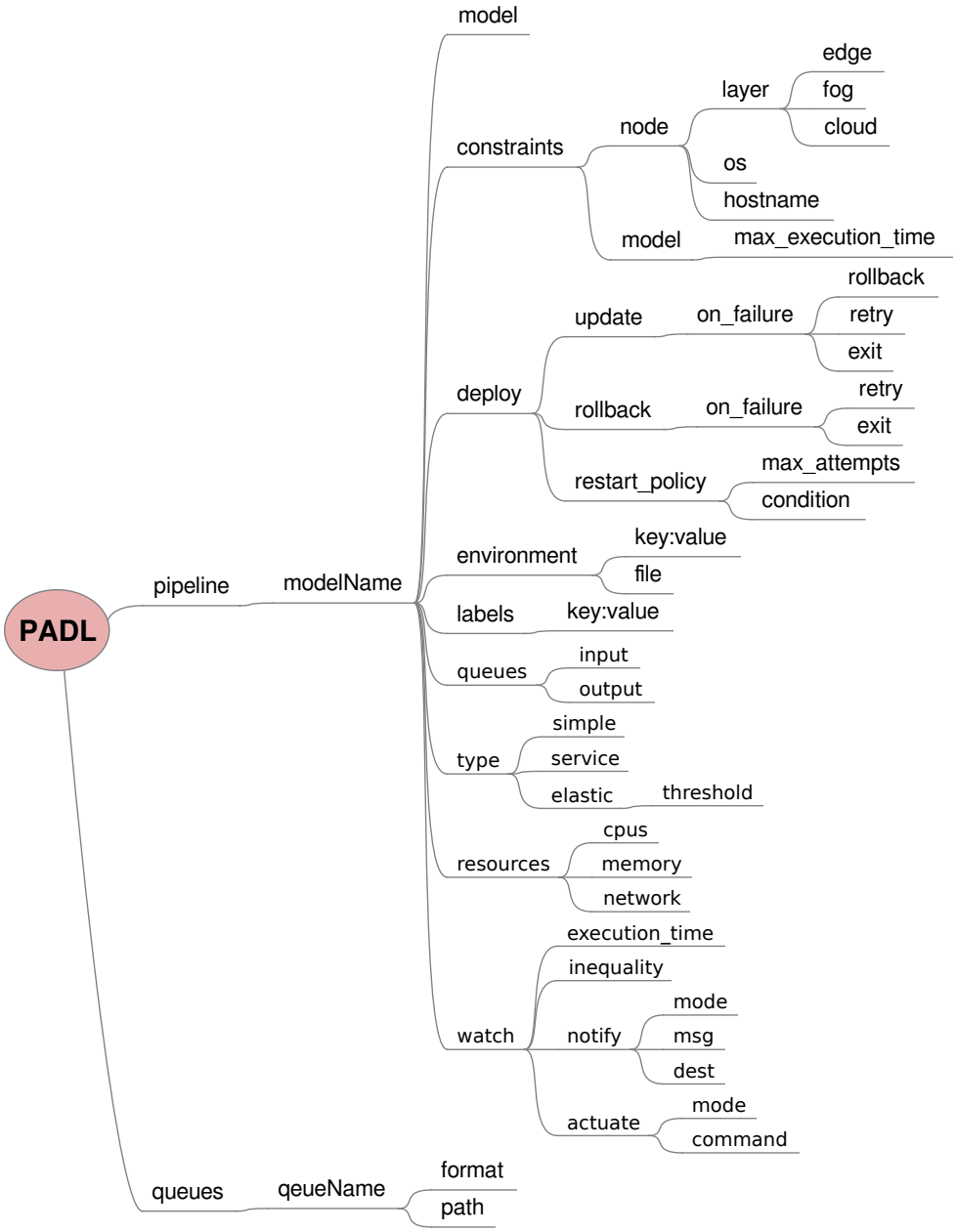

Fig. 3.6 Tree diagram showing the PADL schema.

```
constraints:
  node:
    layer: 'edge'
    os: 'windows'
    hostname: 'slave01.example.com'
  model:
    max_execution_time: 600s
```

Fig. 3.7 Example of the constraints keyword.

```
deploy:
  update:
    on_failure: exit
  rollback:
    on_failure: retry
```

Fig. 3.8 Example of the deploy keyword.

The **type** attribute, which is portrayed in Figure 3.11, defines how a model is deployed in production. Marking a model as simple has no impact, and is deployed once in the infrastructure. Type service, on the other hand, means a copy of the model will be deployed on all the devices that fulfill the specified criteria. Lastly, the elastic type represents a model that will scale up or down based on system load (i.e., if the model is using more than 80% of the system resources, another instance of the pipeline is created and deployed on another device that fully complies with the specified constraints).

A key step when moving an analytical pipeline into a production environment is guaranteeing everything performs under the specified quality constraints. In PADL, this is obtained with the **watch** keyword as seen in Figure 3.12. This keyword is used to define the monitorization by utilizing either language specific annotations such as *execution_time* or model specific Key Performance Indicators (KPI), in which the predicate represents an inequality that should be bigger than or equal to zero.

If any constraint specified within the **watch** keyword is violated, an appropriate action may be specified. Currently, either the *notify* and/or the *actuate* keyword may be used. The former provides a way for sending notifications to the stakeholders when any constraint is activated. Figure 3.13 shows an example of its use.

The latter, which is showcased in Figure 3.14, represents an automatic action taken as a reaction to a constraint violation, such as sending a command through the specified channel.

```
labels:
  node.security: "high"
  node.zone: "europe"
```

Fig. 3.9 Example of the labels keyword.

```
resources:
  cpus: "4"
  memory: "16G"
  network: "10G"
```

Fig. 3.10 Example of the resources keyword.

Both actions can be specified as isolated from one another or combined so, in addition to an action being taken, the stakeholders are notified through the proper channel.

3.4 PADL Implementation

As mentioned above, the deployment of artificial intelligence and machine learning models in production environments is a cumbersome process. This becomes even more obvious in smart city environments due to the variety and heterogeneity of the actors involved, whereas within a single organization, it is easier to make sure everyone is on the same page. Davenport et al. in [62] presented a survey in which they showed that even though investment keeps increasing, advances remain slow in this area. Due to this, this section aims to provide tools for alleviating this challenging process of deploying analytical pipelines in production environments.

Next, in Section 3.4.1, three tools developed for the use of PADL in real-life scenarios are presented. In Section 3.4.2, we showcase how these tools complement each other in the operationalization flow.

3.4.1 Tools

We provide three new tools that comply with the specifications described above. First, in Section 3.4.1, we offer PADLib, a library for facilitating the use of PADL in new and existing projects. In Section 3.4.1, we showcase a Command Line utility for a better integration with continuous integration and delivery pipelines, and in Section 3.4.1, we

```
type: "elastic"
  threshold: "0.8"
```

Fig. 3.11 Example of the type keyword.

```
watch:
  execution_time: 600s
  kpi_1: 0.8 - kpi_1
  kpi_2: kpi_2 - 3
```

Fig. 3.12 Example of the watch keyword.

present Web Lint, which facilitates the use of PADL in the early stages of the development. The source code for these tools is publicly available on GitHub [77].

PADLib

As part of this research, we developed a library to facilitate the use of PADL in AI and ML projects. It is a convenient way to start using the language without having to parse the definition from scratch and provides additional utilities such as the validation of the incoming file and a ready-to-use pipeline object.

The technological implementation of the specifications, described in Section 3.3, was developed using Python 3. The reasoning behind this decision is that Python is a highly regarded programming language in the fields of artificial intelligence and machine learning. This language has evolved from mainly being used for fast prototyping to becoming a fully functional programming language with richly featured analytic libraries (e.g., Numpy ([213]), pandas ([187]), scikit-learn ([56])) that are very appropriate for mature projects.

On the other hand, a PADL document is defined using the YAML format, which is very appropriate for being written and modified manually. However, to the best of the authors' knowledge, there is no mature tool for defining the schema of a YAML document. Given that it is straightforward to transform a JSON document to YAML and vice versa, we utilize JSON Schema [225], which is a vocabulary that allows the annotation and validation of documents, as the preferred method for the description and validation of PADL documents. A snippet of this schema is shown in Figure 3.15.

The entire definition of PADL using JSON Schema can be found in the GitHub repository mentioned above. For the sake of brevity, only a snippet corresponding to the **resources**

```
watch:
  execution_time: 600s
  notify:
    mode: slack_1
    msg: "The constraint $CONSTRAINT_ID$ has been violated"
    dest: ["group1", "group2"]
```

Fig. 3.13 Example of the notify keyword.

```
watch:
  kpi_1: 0.8 - kpi_1
  actuate:
    mode: opc_ua
    command: $CMD_CLOSE_VALVE$
```

Fig. 3.14 Example of the actuate keyword.

keyword is showcased in the listing above. This keyword has been defined as an object, which is comprised of four different entities; all of them are of the *string* type. A violation of any of these definitions would flag an error while trying to validate the document.

This schema combined with Python was used for developing the PADL API. The purpose of this API is to facilitate the use of PADL documents within artificial intelligence and machine learning projects. It provides a validation mechanism for PADL documents and parses the document into a dictionary to be easily consumed afterwards. In order to validate this development, two utilities are released within the same repository, as can be seen in Figure 3.16. A (1) web application (Web Lint), and a (3) Command Line utility (CLI).

Any project willing to integrate PADL into their workflow should follow a similar flow as these two tools. The invocation of the validation method becomes in (2) PADLib validating the document against the schema. Next, the library specifies whether the document is valid, indicating the cause of the error if the validation fails. If it succeeds, it will also return a dictionary with the entities of the document.

Command Line Utility

The first tool developed using PADLib is the Command Line utility. The reasoning behind this tool is maintaining the validity of PADL documents across the development flow. This tool takes a PADL document as an input, and it validates that the file is properly

```
"resources": {
        "id": "#/definitions/resources",
        "type": "object",
        "properties": {
                "cpus": {
                        "type": "string"
                },
                "gpus": {
                        "type": "string"
                },
                "memory": {
                        "type": "string"
                },
                "disk": {
                        "type": "string"
                },
                "network": {
                        "type": "string"
                }
        }
}
```

Fig. 3.15 A snippet of the JSON schema definition for PADL language.

formatted against the schema provided within the API. It solves two different use cases: (i) a user can utilize this tool to manually validate documents, and to this end, it shows an appropriate message if the document is correct or the corresponding error message if the document contains any error; (ii) the application returns a zero or a non-zero value whether the input document is valid or not; hence, it is an appropriate tool to validate analytical pipelines in continuous integration and deployment environments. This paves the way to a more successful deployment of analytical models in production environments.

To this end, we offer a Docker image ([64]) with a stable version of the PADL Command Line utility to be used in continuous integration and deployment environments. In addition, the following snippet is a working example of the tool in conjunction with Travis-CI ([274]), which is a continuous integration and deployment server, and can be integrated into existing CI/CD pipelines. A complete example of the integration of this tool into a CI/CD pipeline can be found on GitHub, and the corresponding example can be found in Figure 3.17.

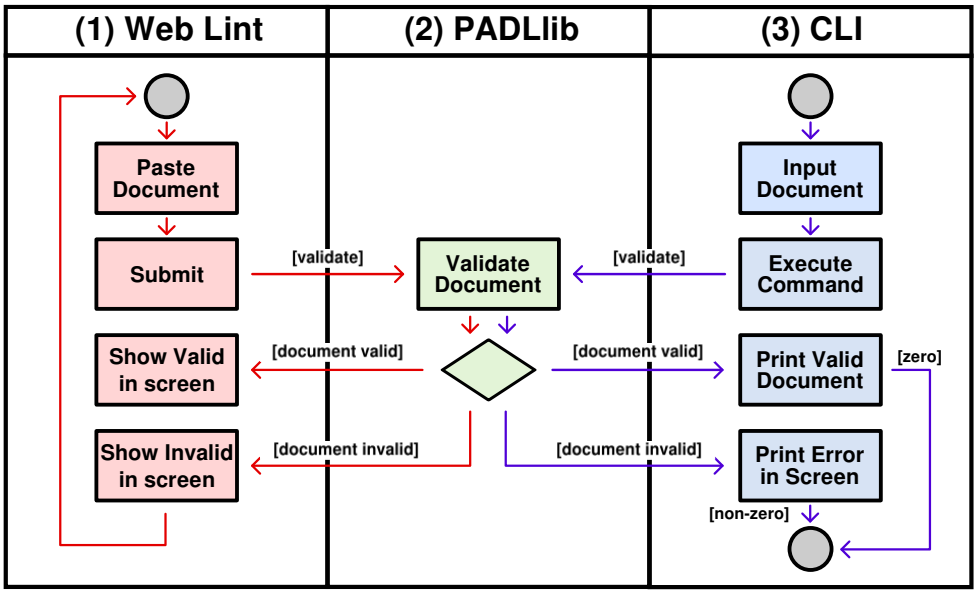

Fig. 3.16 Activity diagram showcasing the interaction of the PADL library with the Web Lint and Command Line (CLI) utilities.

Web Lint

The second tool that uses the API is a web application specifically developed for manually validating PADL documents. We identified the popularity of similar tools such as JSONlint ([156]) and YAMLlint ([303]). These tools have been widely adopted by developers in order to highlight the problems in JSON and YAML documents and lessen the complexity of using these interchangeable formats; hence, we provide an analog utility for PADL. The motivation behind such a tool is that data scientists can be certain that the pipeline they are tailoring is mature enough to be submitted to the development or production environment.

Similarly to the Command Line tool, we provide a Docker image ([65]) that can be used locally. This image can be deployed with the command in Figure 3.18.

Figure 3.19 showcases a screenshot of this tool being used for validating a document. Web Lint is designed to be straightforward to use. A wide text area preceded by a brief introduction in which the pipeline can be introduced covers the screen. Pressing the Go button will yield whether the pipeline is correct with the Valid! message highlighted in green or presents some issues that must be corrected with the Not Valid message highlighted in red.

```
validation:
  image: josuarcaya/padl-cli
  stage: validation
  script:
  - /usr/bin/padl input_file
```

Fig. 3.17 Code snippet representing the validation stage for a PADL document.

```
#!/bin/bash
docker run -p 5000:5000 josuarcaya/padl-web
```

Fig. 3.18 Code snippet for spinning up a PADL web instance.

3.4.2 Delivery Flow

The purpose of the above implementations is to provide a means for the deployment in production environments of artificial intelligence and machine learning pipelines.

Figure 3.20 showcases the interaction of these tools in the analytic development and deployment life cycle. In (a), the data scientist develops and tests the pipeline locally and integrates PADLib in order to utilize PADL within the pipeline. Given that the definition of this pipeline is made by hand, he/she can make use of (1) Web Lint to make sure the definition is correct, without having to interact with any programming language at all. Then, the (2) CLI becomes particularly useful in (b), the continuous integration and delivery pipeline, in order to provide a very high confidence in the production deployment. Finally, in (c), the data scientists can deploy their pipelines in the production environment with the extra confidence of having followed the described process.

3.5 PADL in Cities

In this section, we utilize two use cases from independent authors as validation for the specifications defined in Section 3.3 and the implementation presented in Section 3.4. In Section 3.5.2, PADL serves as the definition and deployment language for the analytical pipeline, and it complements the serious game proposed by the authors. Next, in Section 3.5.3, PADL complements the operationalization scenario the authors propose.

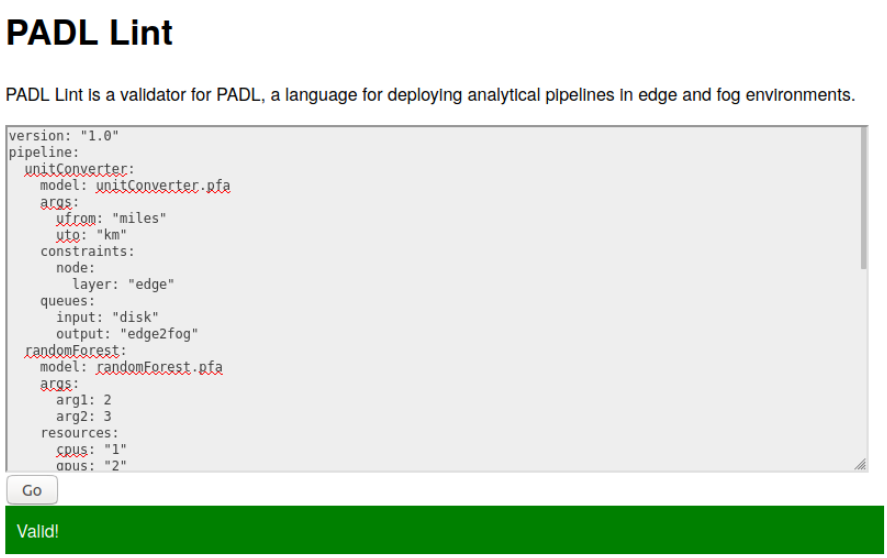

Fig. 3.19 Screenshot of the Web Lint utility validating a PADL document.

3.5.1 Use Case Selection

In this subsection, we explain the reasoning behind choosing the two use cases detailed as application examples, and we introduce the evaluation context, specifying features such as the volume of data handled or the complexity of the information flows. For the selection of these use cases, we leveraged the V's of Big Data (i.e., Volume, Variety, Velocity).

- Flood control: In this use case, the data are produced by many sensors (high volume) measuring the water level spread across the course of the river, all of them gathering data at very low frequencies (high velocity)
- Waste management: This use case includes multiple and heterogeneous sources of information, so the variety and volume of datasets can be the main problems. Among the types of data to consider, we find: city maps, geo-referenced traffic information, garbage collection points, and other aspects such as the social or consumption information of the inhabitants, which can be used for the processes of information crossing, optimization, or calculations of indicators.

Both use cases require complex analytical processes to transform the aforementioned data into information that can assist in the decision-making, which makes them ideal for the validation of PADL.

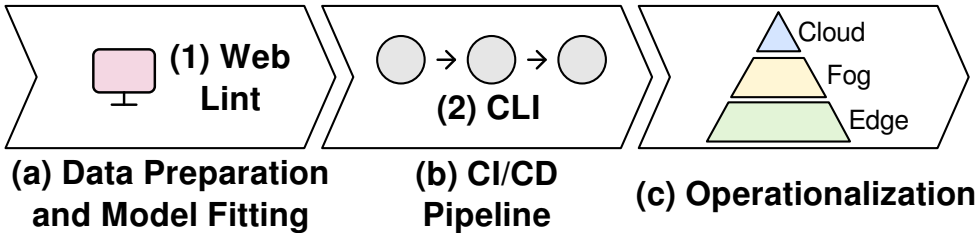

Fig. 3.20 Integration of the PADLib, CLI, and Web Lint tools in the delivery flow.

3.5.2 Flood Control Use Case

In [242], the authors proposed the use of a serious game in the smart city of Prague, which will serve as training for the different stakeholders to react appropriately to a variety of disasters. The scenario is one in which the river Vltava, the longest river in the Czech Republic, overtops its channel's bank and floods Prague, leading to property damage and victims. This is because the level of this river depends heavily on rainfalls taking place during its course. In order to foresee this, the level of the river is measured in different places. In addition, weather forecasting is leveraged, but due to the inherent complexity of doing so over an extended period of time, it becomes very difficult to accurately predict when and where flooding will take place.

This scenario is an example in which PADL excels, as it complements perfectly the serious game detailed in the paper. PADL can be used to describe the monitoring and analytical needs of the use case, and the serious game described will train the relevant stakeholders based on this information. Figure 3.21 showcases an overview of the relevant elements that can be managed and the relationship with the stakeholders. Firstly, the water level of the river Vltava is measured by (1) cyber physical systems installed over its course; should this level rise beyond a predefined threshold, a notification is sent to the relevant stakeholders. The actors in this scenario are defined in the serious game as follows: (i) the management team is composed of the (a) mayor and the heads of the (b) police department and (c) fire brigade; (ii) the observers, which in this scenario are (1) cyber physical systems reporting the water level in different locations, areas getting flooded, and the integrity of the removable dams; and the (iii) first responders, (b) police officers and (c) firefighters and medical staff. Secondly, if an incident (e.g., flooding) preventing a road from being used takes place, notifications are sent to the (c) fire and (b) police departments. Both of these processes operate right where data are generated, in the edge layer, and regardless of their outcome, the information is transmitted to the upper layer. The cloud layer, on the other hand, hosts three different processes. The first one, the (3) weather data service consumes data from a

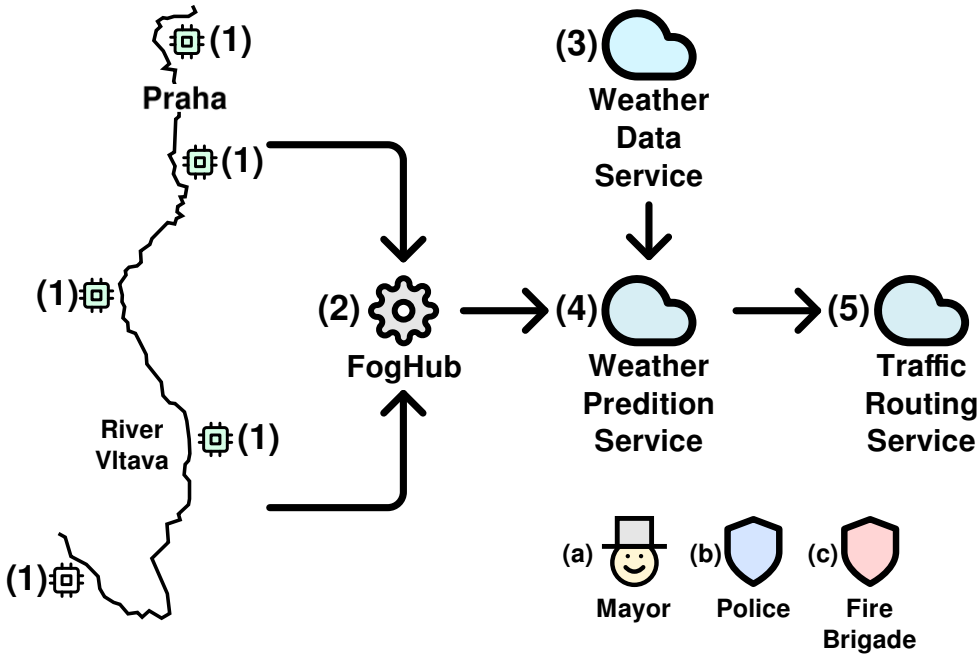

Fig. 3.21 Flood control use case.

weather forecasting service and its duty is twofold: (i) it transforms the collected information to be better consumed by another process, and (ii) it raises alarms based on the incoming data. The second process, the (4) weather prediction service is fed by the (3) weather data service and the data collected in the edge layer. Within this process, a Bayesian model processes the data, and elaborated predictions and higher level notifications are produced. Such predictions can lead to the evacuation of certain areas, road closures, or event cancellations; and certain measures to prevent this such as the building of removable dams can be done in advance. Finally, the (5) traffic routing service collects information and notifications from the Weather Prediction Service and explores different scenarios, in which certain roads are not to be used due to the danger of flooding. This last process is utilized by rescue services (e.g., (b) fire department, (c) police department) in order to execute the assistance in the fastest and most efficient manner. Figure 3.22 offers the above scenario corresponding to the edge and fog layers in detail using the PADL language.

```yaml
version: "1.0"
pipeline:
  waterLevel:
    model: waterLevel.pfa
    type: "service"
    constraints:
      node:
        layer: "edgeRiver"
    watch:
      waterLevel: 0.7 - waterLevel
      notify:
        mode: "email"
        msg: "The water level has raised over the threshold"
  queues:
    input: "waterSensor"
    output: "fog"

  roadState:
    model: road.pfa
    type: "service"
    constraints:
      node:
        layer: "edgeRoad"
    watch:
      roadState: roadState - 1
      notify:
        mode: ["sms", "email"]
        msg: "Please review road $ID$"
    queues:
      input: "roadSensor"
      output: "fog"

  fogNotifier:
    model: fogNotifier.pfa
    type: "simple"
    constraints:
      node:
        layer: "fog"
    watch:
      water_fog_1: 0.7 - water_fog_1
      water_fog_2: 0.8 - water_fog_2
      road_fog: 10 - road_fog
      notify:
        mode: ["email"]
        msg: "Kpi $WATCH_ID$ violated"
```

Fig. 3.22 PADL document definition for the flood control usecase for the edge and fog computing layers.

The *waterlevel.pfa* and *road.pfa* operate in the (1) edge devices and utilize the *watch* entity to monitor and report the data coming from the sensors in case an issue is detected. *fogNotifier.pfa* operates in (2) and is able to aggregate data coming from the edge devices and evaluate more complex scenarios, such as an increase in the water flow at different points of the river. In Figure 3.23 the entities corresponding to the cloud layer are defined.

(3) *weatherProcessor.pfa* is in charge of collecting weather information from an external data source and sending it to the (4) *weatherForecasting.pfa* model. In this service, these data are combined with those generated in the edge and are able to make predictions, so in the case of hazardous scenarios, the relevant stakeholders can be notified (e.g., (b) police

```yaml
weatherProcessor:
  model: weatherProcessor.pfa
  constraints:
    node:
      layer: "cloud"

weatherForecasting:
  model: weatherForecasting.pfa
  constraints:
    node:
      layer: "cloud"
  watch:
    weather_kpi_1: 0 - weather_kpi_1
    weather_kpi_2: 0 - weather_kpi_2
    weather_kpi_3: 0 - weather_kpi_3
  notify:
    mode: "email"
    msg: Kpi $WATCH_ID$ violated"

trafficPrediction:
  model: trafficPrediction.pfa
  constraints:
    node:
      layer: "cloud"
  resources:
    cpus: "32"
    memory: "256G"
```

Fig. 3.23 PADL document definition for the flood control usecase for the cloud computing layer.

department, (c) fire brigade). Finally, *trafficPrediction.pfa* is able to produce alternative routes for the (b) police and (c) fire brigade, so in the event of a catastrophe, they can operate faster and more efficiently.

3.5.3 Waste Management Use Case

Medvedev et al. in [188] leveraged Internet of Things components such as (i) RFIDs, (ii) sensors, (iii) cameras, and (iv) actuators in a smart city environment and proposed an advanced Decision Support System (DSS) for efficient waste collection. This system incorporates data generated by the smart devices and truck drivers in real time and feeds them into a dynamic route optimization model. This model minimizes the inefficiencies of

waste collection within the smart city, by providing alternative routes for the drivers, taking into account traffic jams, inaccessible waste bins, and problematic areas. The final goal of the waste collection system is to improve the quality of service for the citizens of the smart city.

The stakeholders in this scenario greatly benefit from the monitoring and notification capabilities of PADL: the (a) city and district administrations are interested in controlling the process of waste collection, both from a quality of service point of view (e.g., all collected cleanly and in time) and from a legal point of view (e.g., collect evidence for solving disputes); (b) waste truck organizations and drivers want to monitor and track the fleet and find alternative routes based on data gathered from IoT devices; (c) recycling factories want to send notifications based on their current needs and limitations; and the (d) police department needs to be notified if improper parking is preventing the waste bins from being emptied. These entities alongside the services interacting in this use case are represented in Figure 3.24. Some of these entities feed information into the system. First, Entity (b), waste trucks, use IoT devices to report their location, capacity, and fuel in real time. In addition, the drivers are able to report problems (e.g., improper parking) while collecting waste bins. This information includes video and audio information that the drivers are able to publish by using mobile devices. Entity (c), recycling factories, are able to publish their capacities or needs based on their storing or recycling desires. Finally, the ecosystem is able to process information coming from surveillance cameras and traffic and weather services. Three services comprise the analytical engine of the ecosystem. Firstly, the (1) Tracking and Monitoring Service is able to provide metrics and notifications to the *(b) Waste Organizations* so they are provided with advanced insights about their business to make better decisions. Secondly, the *(2) Surveillance Service* uses the data generated by surveillance cameras and notifies (a) the *City Administration*, (b) truck drivers, and (d) police department in case a problem (e.g., traffic jam, blocked road) is detected. Finally, the *(3) Traffic Routing Service* is able to provide dynamic routes to the (b) truck drivers based on all the information being fed into the system, hence minimizing the time needed to perform their duties. There are two entities (the *(a) City and District Administrations* and the *(d) Police Department*) that do not feed the ecosystem with data, but only interact with it by receiving notifications. The former wants to be up to date on the whereabouts of the waste recollection process in the city, whereas the latter will react to the notifications being sent by the *(2) Surveillance Service* and the *(b) Truck Drivers* in order to solve the issues being raised. Figure 3.25 defines the above scenario using the PADL language.

The *cameras.pfa* model operates in the edge devices and is able to notify about blocked roads to the (d) Police Department. In addition, it serves as an input for the (2) *Surveillance Service*, which performs a wider aggregation of the data generated by the cameras and is able to provide high level notifications such as traffic jams. The (1) *trackmon.pfa* model operates

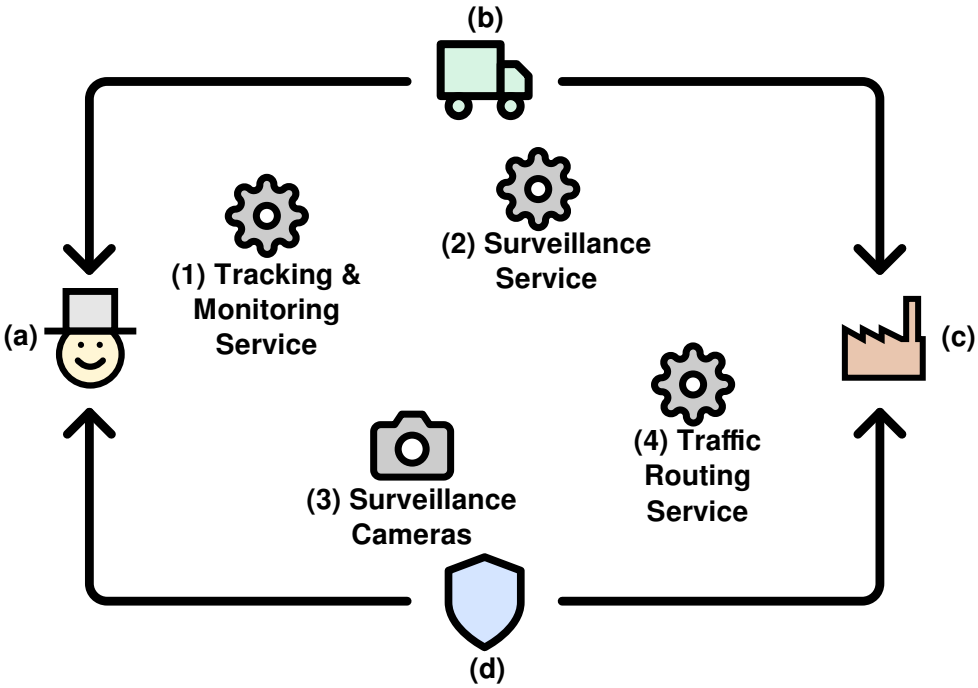

Fig. 3.24 Waste management use case.

in the cloud and is able to get the information produced by track drivers. Finally, the (3) *routing.pfa* provides alternative routes based on all the data and insights generated by the ecosystem.

3.5.4 Use Case Discussion

This section presents the main results of the process to validate the proposed language by means of both use cases: flood control and waste management. In order to trust in the feasibility of the PADL language to represent analytic pipelines and to operate them in production environments, it is relevant to compare the definition file provided in each use case against the criteria in Table 3.2. In addition, a seventh criterion is analyzed regarding the complexity of the use case measured through its characterization through the three V's of Big Data: volume, variety, and velocity. This analysis can be seen in the following table, where the use cases are presented as the rows and the evaluation criteria form the columns.

Based on our observations of these use cases' PADL implementations, we were are encouraged by the results, showing that:

```yaml
version: "1.0"
pipeline:
  cameras:
    model: cameras.pfa
    type: "service"
    constraints:
      node:
        layer: "edgeCamera"
    watch:
      roadBlocked: 0.1 - roadBlocked
      notify:
        mode: "sms"
        msg: "Road $ROADID$ may have been blocked"
        dest: ["police", "council"]
    queues:
      input: "camera"
      output: "surveillance"

  trackmon:
    model: trackmon.pfa
    constraints:
      node:
        layer: "cloud"
    watch:
      fuel: fuel - 0.1
      capacity: 0.9 - capacity
    notify:
      mode: "sms"
      msg: "Notification $NOTID$ has been raised: $MSG$"
      dest: ["police", "council"]
    queues:
      input: "trucks"
      output: "traffrout"

  surveillance:
    model: surveillance.pfa
    constraints:
      node:
      layer: "cloud"
    queues:
      input: "surveillance"
      output: "surveillance_b"

  routing:
    model: routing.pfa
    constraints:
      node:
      layer: "cloud"
    queues:
      input: ["trafrout", "surveillance_b"]
      output: "db"
```

Fig. 3.25 PADL document for the waste management use case.

1. Criterion (1): We provide domain specific annotations for the appropriate deployment of each step in the pipeline, such as the measurement of the water level in the river in the first use case and being able to detect blocked roads with the cameras in the second one.

2. Criterion (2): Monitorization, notification, and actuation are handled separately for each model. For example, in the second use case, the language enables the chance of monitoring and reporting a low fuel level for the waste truck.

3. Criterion (3): We utilize technologies and smart devices already present in the use cases, like the ones used for reporting the river level in the water flood use case.

	Deployment Awareness	Analytics Oriented	Technology Agnostic	Text Based	Small Technological Footprint	Multilayer Awareness	Big Data Dimensions
Flood Control	✓	✓	✓	✓	✓	✓	Volumen and Velocity
Waste Management	✓	✓	✗	✓	✓	✓	Volumen and Variety

✓: validated; ✗: not validated.

Table 3.2 PADL functionalities following the criteria of Table 3.1 validated against the use cases.

4. Criterion (4): All the definitions of the use case analytic pipelines are covered in the PADL snippets in Sections 3.5.2 and 3.5.3.

5. Criterion (5): We do not require additional technologies other than the ones already in use by the city. For example, in the case of waste management, cameras are the smart devices that are mainly used.

6. Criterion (6): The different steps of the pipeline are deployed across the existing infrastructure and can be operationalized independently. For example, in the waste management use case, analytic processing is distributed over the heterogeneous infrastructure: waste trucks, cameras, cloud servers.

7. Criterion (7): The main dimensions of Big Data, volume, velocity, and variety, are highly represented in the use case. In the flood control use case, the most stringent restriction is the speed of data collection and analysis for the flooding alerts in real time. In the waste management use case, on the other hand, the volume and heterogeneity of the data sources are the greatest challenges, due to the need to cross-reference information to draw valuable conclusions that help optimize waste management.

In conclusion, PADL is an appropriate solution for the smart city domain, advancing over existing tools for similar purposes as per the evaluation criteria in Table 3.1. The previous analysis shows that PADL excels in defining complex analytical pipeline flows, with multiple considerations (heterogeneous data sources, multiple devices to consider, or different types of processing) and various phases (e.g., monitorization, notification, actuation).

3.6 Conclusions and Future Work

In this research, we propose a domain specific language for the definition of distributed analytical pipelines, which alleviates the burden of deploying data science projects in production environments. On one hand, it abstracts data scientists from the underlying technologies,

networking challenges, and deployment specific constraints, hence letting them focus on the functionalities. On the other hand, the team in charge of the operationalization obtains a detailed description of the process for deploying the given pipeline in the production environment. Overall, PADL aims at raising the success rate of data science projects, by mitigating the challenges of deploying such projects in real-world scenarios. The use of modern mechanisms to provide these pipelines with monitorization, notification, and actuation capabilities promote the utilization of this language in the emerging IoT paradigm. In order to do so, we provide tools that cover the operationalization of ML (Machine Learning) and AI (Artificial Intelligence) projects over its different stages and lighten the burden of deploying these projects in edge, fog, and cloud environments successfully. As part of this effort, we analyze two use cases from independent research papers in a smart city context and elucidate how PADL can be used to implement the analytical needs that complement them in order to provide an integrated solution. Both use cases demonstrate the benefits that PADL offers in the smart city domain, and unlike the rest of the alternatives under study, it has been shown to comply with all the criteria defined in Table 3.1. The source code with the language specification and the developed tools are provided by Díaz-de-Arcaya et al. in [77].

As for the future work, we consider two research directions: firstly, to promote PADL as the standard for the definition of analytical pipelines leading to the development or evolution of other tools or utilities; secondly, to increase the expressiveness of PADL with new functionalities and characteristics spanning a broader range of the analytical life cycle. In the first line of study, our idea is to utilize PADL alongside the definition of a given infrastructure to implement the orchestrator appearing in Figure 3.1; in this way obtaining the best possible deployment for a given analytical pipeline. Furthermore, the emergence of public cloud providers in conjunction with infrastructure-as-code technologies gives us the possibility of generating the necessary infrastructure on demand.

Instead of needing lots of children, we need high-quality children.

Margaret Mead

4

Operationalization Framework for Distributed Pipelines

The work presented in this chapter was published in 2023 in the Future Generation Computer Systems journal [80], which received an impact factor of 7.5 in 2022 and was ranked as a Q1 in JCR. This chapter delivers TC4, TC5, SC5; therefore fulfilling O3 and O4.

The use of Artificial Intelligence solutions keeps raising in the business domain. However, this adoption has not brought the expected results to companies so far. There are several reasons that make Artificial Intelligence solutions particularly complicated to adopt by businesses, such as the knowledge gap between the data science and operations teams. In this paper, we tackle the operationalization of distributed analytical pipelines in heterogeneous production environments, which span across different computational layers. In particular, we present a system called Orfeon, which can leverage different objectives and yields an optimized deployment for these pipelines. In addition, we offer the mathematical formulation of the problem alongside the objectives in hand (i.e. resilience, performance, and cost). Next, we propose a scenario utilizing cloud and edge infrastructural devices, in which we demonstrate how the system can optimize these objectives, without incurring scalability issues in terms of time nor memory. Finally, we compare the usefulness of Orfeon with a

variety of tools in the field of machine learning operationalization and conclude that it is able to outperform these tools under the analyzed criteria, making it an appropriate system for the operationalization of machine learning pipelines.

4.1 Introduction

Over the last few years, companies have been adopting artificial intelligence (AI) solutions as a basic disruptive technology to obtain competitive advantages in their businesses or even discover new opportunities for success [257, 291, 104]. There are many promises that AI-based solutions aspire to materialize, in [67] the authors outline the most important benefits of utilizing data science in businesses: support for data analysis and insight generation with agility, decision-making based on data-derived facts, improvement of data quality, facilitating the understanding of the business environment, and opportunity sensing and organizational performance management. On the other hand, the authors identify as well the most important challenges: data science training, allocation of investments in analytical technologies, and data governance and strategy. Several of these challenges can be alleviated by implementing a correct process for the operationalization of AI in business environments. Accordingly, traditional human-dependant development and maintenance need to transform to Artificial Intelligence for IT Operations (AIOps), in which the use of genetic algorithms has attracted attention [315]. For instance, genetic algorithms have been utilized for QoS optimization within AIOps [186].

This article is framed in the context of AI operationalization, understanding it as the deployment of AI models into real-world production environments. It is a complex stage of the AI process, due to the high heterogeneity of both models and environments. Hence, the knowledge necessary for the operationalization to be executed correctly, is extensive and involves experts from different fields. This problem is tackled in detail in the following sections, alongside the contributions of our work that we propose to mitigate it and that we present in this article.

4.1.1 Problem Statement

At the moment, data science and software engineering are largely decoupled, and incorporating machine learning inside applications is costly and difficult [3]. In addition, machine learning operationalization requires high expertise and cross-domain knowledge [110, 7], and it is often intended to maintain a reliable performance operating in harsh conditions [169, 286]. In this paper, we try to narrow this gap between the data science and operations teams of any given organization. The former is very often unaware of the particularities of deploying

their workloads in production environments (e.g., infrastructural devices, solution packaging, networking), whereas the latter lacks the experience in machine learning for properly deploying such solutions effectively. The majority of data scientists are not computer scientists by training, and companies that have embraced a data strategy are short of data engineers [266]. In fact, it is important for data scientists to be conscious of the environment in which their ML model will be eventually deployed (e.g., if it becomes operationalized in a resource-limited environment, if there are time constraints, or how the model will fit with the broader software). All these need to be considered for a successful operationalization of a data science project [175]. Due to this complexity, it happens very often in systems that utilize machine learning methods to end up with high technical debt, so it is important for both engineers and researchers to be aware of it [250]. In order to alleviate this, there are a series of best practices related to machine learning in software engineering in which we can highlight end-to-end pipeline support; and model evolution, evaluation, and deployment [10]. Moreover, the lifecycle of ML applications differs from that of traditional software components [167]. Actually, the ML lifecycle is more complex than traditional software development, hence minimizing waste, and accommodating early changes becomes crucial [158]. In this regard, Machine Learning Operations (MLOps) has gained traction as it promotes the collaboration and communication between data scientists and operations professionals [232]. In particular, MLOps refers to the process of automating all the steps of the ML life cycle, including development, integration, testing, releasing, deployment, and infrastructure management [183]. The focus of this manuscript is on these last three stages.

4.1.2 Contribution

After analyzing the inherent difficulties of the aforementioned problem, in this article we offer the Orfeon AIOps framework. It focuses on the operationalization of distributed analytical pipelines in heterogeneous environments. The main contributions of this platform are the following:

C1 The proposed framework focuses on the operationalization of data science projects to raise the success rate and adoption of such endeavors.

C2 The metric shipper, which operates in the cloud and edge layers, is delivered as part of this research. It feeds Orfeon with the necessary metrics to operate on both newly created and existing infrastructure.

C3 We provide a framework that operates seamlessly across heterogeneous distributed infrastructural devices, and is able to utilize them optimally. In particular; it can

leverage different devices in the cloud, edge, and on-premises computational layers; operating under different platforms.

C4 The mathematical formulation of the system for the goal-driven deployment of an analytical pipeline is explained in detail. Specifically, this research leverages the cost, performance, network, and privacy of the solutions; all subject to a series of constraints.

C5 We offer a framework that integrates with the existing technological stack. For instance, Orfeon does not aspire to replace other solutions already in use in the different stages of the machine learning deployment. Instead, it offers goal-driven operationalization capabilities on top of them.

The rest of the paper is organized as follows. Section 4.2 offers a brief introduction of the background. In Section 4.3, the related work is showcased and a comparison criteria is established. The overview of the system is presented in Section 4.4 and the optimizer itself is explained in further detail in Section 4.5. Section 4.6 portrays the experimental results. An expert evaluation of Orfeon is found in Section 4.7, followed by a discussion in Section 4.8. Finally, the conclusions and future directions are drawn in Section 4.9.

4.2 Background

The increase of data being produced at the edge of the network has led to previous paradigms such as cloud and edge computing being no longer efficient [252]. In addition, the many benefits introduced by edge computing such as speed, security, scalability, versatility, reliability [117]; gain the attention of other technologies such as Edge AI, which promotes lowering artificial intelligence models down to the edge layer.

Traditionally, the operationalization of artificial intelligence solutions in production environments has proven to be a cumbersome process. In order to tackle this problem, the declarative abstraction of machine learning models and analytical pipelines is a previous step towards the successful operationalization of such projects. In this regard, the Data Mining Group developed the Predictive Model Markup Language [229], which is a predictive model interchange format based on XML. Pivarski et al. describe in [227] an extensible language for the description of mathematical, statistical and machine learning models called Portable Format for Analytics (PFA). Its purpose is to abstract the analytical models from the underlying tools, application, and systems. Similarly, Dou et al. in [83] narrows the focus on purely genetic programming trees, and propose a language entitled GPML for standardizing the definition and sharing of such models. One step further was taken in [79], in which the authors tried to address the problem of deploying distributed analytical pipelines over

different computational layers. In [199] the authors present a distributed framework that unifies the training, simulation, and serving of reinforcement learning applications. They introduce the concept of agents that repeteadly interact with the environment to maximize a reward.

On the other hand, the Python language [281], very often regarded as a fast prototyping language, has a wide variety of libraries for data analysis, processing, and visualization. This makes it appropriate for optimization problems. There is a wide range of optimization frameworks that implement different algorithms ready to be used. For instance, the Gurobi optimization framework [184] is a proprietary solution supporting a wide range of programming languages (e.g., C++, Java, .NET, Python). Blank et al. presented Pymoo in [27], an open source multi-objective optimization framework developed in Python, which supports decision-making and the visualization of the solutions. Another framework for problem optimization is Platypus [228] also implemented in Python, which offers similar capabilities, but does not support visualization. From the well regarded multi-objective optimization framework jMetal, originally developed in Java [86], originated jMetalPy [23], its Python implementation. This framework is built on the experiences of jMetal and takes advantage of the Python language.

Finally, the optimization of different problems is a key topic in academia. For instance, in order to find the optimal deployment for different scenarios, the authors propose in [195] a hybrid algorithm to resolve the problem of 3D indoor deployment in wireless sensor networks (WSN). Similarly, in [295] the authors propose an algorithm for urban WSN deployment. In [313] the authors propose a methodology to optimize the resource allocation model for solving crowd-based cooperative task allocation problems in a cost-efficient and requirement adapted way. Ma et al. propose in [181] a knowledge-driven evolutionary algorithm for the deployment and startup of microservices, they utilize multi-objective optimization but focus solely on these microservices. The adoption of cloud computing technologies is also under extensive research. Arostegui et al. propose in [13] a methodology to optimize the definition of Infrastructure as a Service cloud models for a Big Data platforms. It utilizes evolutionary heuristics for finding the optimal configuration. Frey et al. present in [98] a simulation-based genetic algorithm that optimizes cloud deployment options for supporting the migration of software to the cloud. Given that the pricing of cloud solutions becomes harder due to the amount of possibilities, in [51] the authors provide a novel mathematical approach that alleviates having to deal with the heterogeneity and pricing models of these cloud services. In order to do that, they present a queuing theory based Mixed Integer Linear Program to find a multi-cloud configuration. In [131] the authors present an iterative mathematical decision model to be used by organizations to evaluate the decision of investing on on-premises infrastructure or outsourcing to multicloud environments. Examples of optimization in

the networking domain can also be found, in [298] the authors propose a novel strategy that allocates reserved bandwidth for cyber-physical systems. This way, they are able to minimize the operating cost and the influence of the discrepancy between SLA (service level agreements) and the network resources while preserving QoS (quality of service) level.

4.3 Related Work

The automation of the machine learning life cycle is the cornerstone for raising the success rate of artificial intelligence projects in industry. There are a wide variety of tools that address various stages of this machine learning life cycle, including the operationalization of such projects in production environments. Table 4.1 offers a comparison of many of these technologies, and evaluates them against the following features:

F1 Machine Learning: it has been specifically tailored to support machine learning projects and understands its nuances.

F2 Pipelines: it supports not only isolated models but also complete analytical pipelines.

F3 Technology Agnostic: it has a low technological footprint on the devices, and it integrates seamlessly with other frameworks.

F4 Distributed Infrastructure: it integrates seamlessly with existing distributed infrastructure, and is able to utilize it optimally.

F5 Scalability: it can function optimally increasing its performance in accordance with the demand.

F6 Open License: it is available under an open license and does not enforce propietary solutions to be used.

F7 Goal-Driven Deployment: it aids in the deployment of workloads in production environments, leveraging different objectives.

These features have been selected from scientific literature as they represent significant challenges that we address in this research. In [247] the authors list a series of challenges in the management of machine learning models, amongst which we highlight the difficulty of defining an ML model (F1), the lack of a declarative abstraction for the whole ML Pipeline (F2), and the manifold frameworks and tools coexisting in ML projects (F3). In addition, distributed Machine Learning is becoming the norm over single-machine solutions, and further studies on the performance and scalability are required [284] (F4, F5). The reasoning

	Machine Learning	Pipelines	Technology Agnostic	Distributed Infrastructure	Scalability	Open License	Goal-driven deployment
ModelKB [103]	✗	≈	✗	✗	✗	✗	✓
DLSpec [58]	≈	≈	✗	✗	✗	✗	✗
MLflow [306]	✗	✓	✗	✗	≈	✓	✓
CodeReef [100]	✗	✓	✗	✗	≈	✓	✓
FlexServe [285]	✗	✓	✗	✗	≈	✓	✗
TensorFlow-Serving [214]	✗	✓	✓	✓	≈	✓	✗
ModelHub [190]	✗	≈	✗	✗	✗	✓	i
Kedro [37]	✓	✓	✓	✓	≈	✓	✓
Kubernetes [95]	≈	✗	✗	✓	≈	✓	✗
Kubeflow [63]	≈	✓	✓	✓	≈	✓	✗
Katib [317]	≈	≈	✗	✓	≈	✓	✗
GoalD [241]	≈	✗	✗	✓	✓	✓	i
ApacheBrooklyn [94]	✓	✗	✗	✓	≈	✓	✓
Spinnaker [259]	✓	✗	✗	✓	≈	✓	✓
WatsonStudio [135]	≈	✓	✓	✓	≈	✗	≈
SageMaker [8]	≈	✓	✓	✓	≈	✗	≈
AzureMachineLearning [191]	≈	✓	✓	✓	≈	✗	≈
Michelangelo [127]	≈	✓	✓	✓	≈	✗	≈
GoogleAIPlatform [105]	≈	✓	✓	✓	≈	✗	≈
Orfeon	✓	✓	✓	✓	✓	✓	✓

✓: full support, ≈: partial support, i: not enough information, ✗: not supported

Table 4.1 Tools that support the deployment of machine learning projects in production environments.

behind positively valueing an open license (F6) project is due to the fact that they are easy to use and build upon [91], hence they become easy to access and utilize for both research [243] and industry. Finally, we devote to AIOps as a manner to boost engineering productivity, and reduce operational costs [60] by promoting the auto adaptation of the system. Due to this, we utilize goal-driven algorithms to solve the problem of machine learning operationalization (F7).

The machine learning life cycle differs from the utilized in software development. However, being able to support its different stages and automate it becomes key for the success of such projects. In this line, ModelKB [103] is a tool designed to automate the model management process with minimal user intervention, while letting the user utilize its favorite framework. However, it only focuses on deep learning and not in machine learning as a whole. In addition, it does not cover the pipeline definition and the deployment phases, nor it is aware of the underlying infrastructure. DlSpec [58] is also focused on deep learning, it specifically addresses the fact of not having a standard specification for sharing, running, reproducing, and comparing the innovations being made in deep learning. The authors offer a hardware agnostic deep learning specification, but do not dive deep into the deployment, or scalability concerns.

MLflow [306], on the other hand, is designed to encourage data scientist to follow engineering processes to raise their projects commercial impact. However, it does not focus

on scalability nor does it explore the available infrastructure. CodeReef [100] promotes the deployment of ML models across heterogeneous systems, while it does ease the deployment of such models, it does not cover the concept of pipelines, nor it utilizes existing infrastructure and the scalability issues that may arise. TensorFlow-serving [214] is an open-source system to serve machine learning models. It is production ready and a performance analysis have been performed by the authors. However, it is not aware of the underlying infrastructure, and the deployment it offers is not goal-driven. FlexServe [285] does a similar task, but without the error-prone process of converting the models into static computational graphs. To the best of the author's knowledge it does not leverage the infrastructure, nor it covers pipeline definition, or scalability. ModelHub [190] focuses solely on deep learning, in particular on how to store, share, and reuse such models. It does not cover deployment, pipelines nor it explores scalability concerns. Finally, Kedro [37] is an open-source framework that promotes the use of software engineering best practices in data science projects. It has built-in support for distributed deployment in a variety of platforms (e.g., Docker, AWS, Kubeflow, Databricks), but it relies on other libraries to perform this deployment, hence it lacks a goal driven approach to resolve this issue.

GoalD [241] is a framework which tackles autonomous deployment in heterogeneous and dynamic computing environments. It fully embraces goal-driven deployments, but it does not yet support distributed infrastructural elements, nor it focuses on machine learning workflows. Apache Brooklyn [94] and Spinnaker [259] both tackle the problem of multi-cloud continuous delivery of applications. They are open-source tools dedicated to the release of software changes reliably, but they approach the problem in a general manner and lack the granularity to utilize machine learning projects and analytical pipelines.

While Kubernetes [95] on itself can fulfill many of the requirements specified, it is a general purpose framework, and does not adhere to machine learning specifically. In addition, to be able to fully utilize the infrastructure it requires a heavyweight configuration on the worker nodes. On the other hand, many authors [275, 254, 132] have proposed the use of Kubernetes for the deployment of artificial intelligence projects. In addition, there has been an upsurge of technologies leveraging Kubernetes that focus on machine learning. In this respect Kubeflow [63] arises as a promising framework built on top of the many benefits of Kubernetes, and focuses on lightening the burden of deploying machine learning workflows. Katib [317] is a scalable Kubernetes-native platform. Even though it provides deployment capabilities, these can not be optimized based on goals. It supports only a range of AutoML algorithms, and does not include pipeline definitions.

Up to this point, the analysis has been made over existing open-source tools already in industry, and tools found in academia. However, major cloud providers all implement their own solutions for assisting in the machine learning operationalization. In this regard,

IBM Watson Studio [135] helps in building, running, deploying, and managing AI models. Similarly, Amazon SageMaker [8] helps data scientists and developers to prepare, build, train, and deploy machine learning models. Azure Machine Learning [191] promotes end-to-end MLOps, and interoperability by integrating with open source frameworks assisting in the ML operationalization. Uber's Michelangelo platform [127] is a ML-as-a-service platform designed to meet their business needs. However, unlike the other platforms mentioned, it can not be utilized outside Uber. Similarly, Google embraces MLOps with AI Platform [105]. All these products excel at managing the machine learning operationalization, and offer tremendous scalability. However, they are all proprietary solutions and introduce the risk of becoming dependant on a specific platform, not being able to easily transition to other solutions without incurring heavy expenses. In addition, they lack the ability to perform deployments based on predefined goals.

In summary, tools that focus on the definition of machine learning models are key to the operationalization of analytical models, but they do not embrace the capabilities to efficiently deploy such models in production environments. A wide variety of frameworks cover the machine learning operationalization, but they do not leverage existing infrastructure, nor do they perform this deployment based on predefined goals. Similarly, kubernetes based solutions focus on encapsulating ML models inside containers, but they depend on a kubernetes cluster, hence integrating existing infrastructure is not straightforward. Frameworks that cover the continuous delivery of applications are not yet mature to handle the differences between ML projects and traditional software projects. Finally, major cloud providers have developed their own platforms covering the ML operationalization, but to the best of the authors' knowledge, none cover the deployment of such applications based on predefined goals. Based on the information gathered so far, in this paper we present Orfeon, an ML focused framework that is able to utilize existing distributed infrastructure for the deployment of analytical pipelines leveraging predefined goals. In addition, it integrates seamlessly with other frameworks that cover distinct stages of the ML life cycle.

4.4 System overview

In this section, an in depth explanation of the main elements and stages of the Orfeon operationalization flow is described in Subsection 4.4.1. Next, the two main entities of Orfeon are explained in detail: the metric shipper 4.4.2, and the core system 4.4.3. The former is in charge of gathering the necessary information for the latter to yield and optimal deployment. The implementation has been done in Python 3 due to being cross-platform and being packaged by default in a wide variety of operative systems.

4.4.1 Workflow

There are a series of stages necessary for Orfeon to yield and appropriate deployment definition, these are depicted in detail in Figure 4.1. First, an analytical pipeline needs to be defined. There are several alternatives that would provide Orfeon the capability of interacting with analytical pipelines (e.g., Pachyderm [145], Valohai [280], DVC [146]). In this research we have utilized PADL [79] format, which promotes the fifth contribution (C5) stated in Section 4.1.2, since it builds its capabilites on top of other technologies already in use by the deployment. This definition contains the necessary information for Orfeon to handle the different elements of the analytical pipelines in production environments. It comprises the constraints for each model, its interaction with the environment and other models within the pipeline, privacy concerns to be considered during the deployment, and monitoring capabilities, among others. This element needs to be filled out by both the data scientist and the data engineer, the former elaborates the details regarding the analytical pipeline, and the models it is comprised of; whereas the latter is able to provide information about the technical aspects of the deployment such as the hardware requirements or the privacy constraints. The described analytical pipeline comprises a series of models that need to be recovered in later stages. Orfeon aspires to utilize already established technologies for the serialization [234, 93, 96] and versioning [178, 143] of such models. This document is then fed into the system for further processing. Given that Orfeon has been designed to be as easy to use as possible to facilitate its widespread adoption, the second element, the enriched infrastructure description file, is filled automatically by the metric shipper already deployed on each and everyone of the infrastructural devices, and the information gathered is stored in a centralized database for Orfeon to query when necessary. In order to alleviate the connectivity issues of having to communicate distributed infrastructural devices in a central database, we have utilized a PostgreSQL [196] deployed in AWS [9], since it provides the possibility of having a fully qualified domain name accessible worlwide for the metric shipper and the Orfeon core. This process is described in further detail in subsection 4.4.2. Next, the Optimization stage consists of two different elements: the multiple objectives, and the deployment optimization. The former provides Orfeon with the ability to optimize deployments based on a set of predefined goals. In this paper four objectives have been selected: i) cost, ii) security, and iii) performance. These objectives are formalized in further detail in Section 4.5.1, and have been implemented in GitHub [74]. Even though for this research the aforementioned objectives have been utilized, the system has been designed with the mindset of facilitating the implementation of further objectives easily. Then, the Orfeon deployment optimization evaluates these three elements: the i) PADL defined analytical pipeline that has been manually formalized, the ii) enriched infrastructure

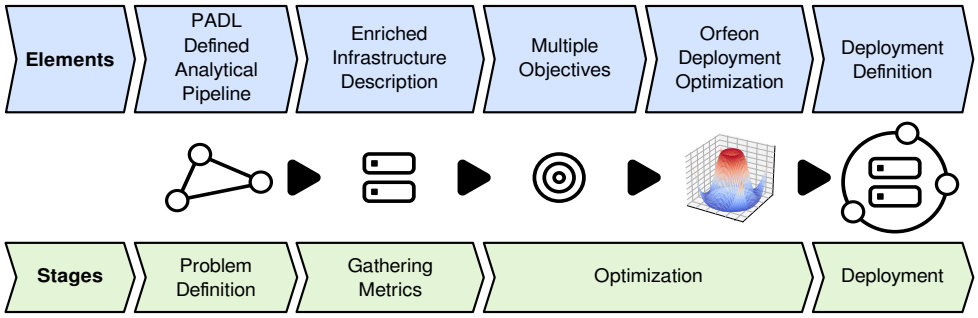

Fig. 4.1 Orfeon goal based analytical pipeline deployment flow.

description gathered by the metric shipper, and the iii) multiple objectives which are already implemented in the Orfeon codebase; and yields a deployment definition in which each device of the infrastructure is assigned with a set of models from the input pipeline leveraging the predefined objectives and constraints. The translation of this deployment definition into an actual deployment is out of the scope of this research, and Orfeon delegates on well established technologies such as Function-as-a-Service (FaaS) [237, 236, 215] and Platform-as-a-Service (PaaS) [125, 300] solutions.

4.4.2 Metric Shipper

The metric shipper is a daemon that integrates with every device that conforms the infrastructure, and collects the different metrics to be used by the core system. There are technologies commonly utilized for gathering data from distributed infrastructural devices [144, 35]. However, for this research we have utilized Orfeon's metric shipper to obtain custom information such as machine learning performance, security, and comsumption. Table 4.2 highlights the different modules in use by the metric shipper, the type of module and the set of tools utilized by it to gather the information.

The main goal is to gather information about the system, and send it to a central repository in a secure and reliable way. It performs this task through two different implementations, the System Information, and the Benchmark modules. The first one facilitates the development of tools that query the system to get basic information about it. In particular, the number of threads the system can run in parallel, the amount of memory it can allocate, the capacity of the installed GPU, or the geographical location in which a particular device resides. The kind of jobs that implement the System Information module are lightweight and rarely need the use of external resources. They rely solely on tools already present in the system and finish their task rapidly. The metrics gathered in this module are mainly used by the core system to

Name	Type	Tools
CPU	System Inf.	/proc/cpuinfo
Memory	System Inf.	free
GPU	System Inf.	lspci
Location	System Inf.	IPWhois [120]
Consumption	Benchmark	powerstat
Resilience	Benchmark	lynis [53]
Performance	Benchmark	AI-Benchmark [139]
Parallelization	Benchmark	stress-ng [262]
Latency	Benchmark	tcp-latency [176]
Throughput	Benchmark	speedtest-cli [44]

Table 4.2 Metrics recorded by the metric shipper in each of the managed devices and their associated tools.

evaluate constraints (e.g., no device can be assigned a model which allocates more memory than the already present in the device, a model cannot be deployed in a geographical location that violates its privacy concerns).

On the other hand, the benchmark module is in charge of doing the heavy lifting when assessing a system. Tools developed under this module very often require external dependencies and tools, and can take a long time to finish. In this paper, we have implemented four benchmarks. The first one, the consumption benchmark, stresses the CPUs and GPUs of the system while taking consumption measurements of the different configurations using the stress-ng [262] tool. The second one, the resilience benchmark, utilizes the well known security audit tool Lynis [53], which is widely used both in industry and in academia [113, 205, 212]. It has proven to be very convenient as it scans the security aspects of the system and provides a final score to it. Thirdly, the performance benchmark evaluates how the system performs when executing artificial intelligence workloads with AI-Benchmark [139]. This open source tool evaluates the AI performance of the platform relying on the Tensorflow machine learning library, and is actively used in academia [141, 39]. The parallelization benchmark is able to asses the performance penalty the system undergoes when executing several threads at the same time. It does so by stressing the system and measuring the response time under different configurations by using the stress-ng tool. Finally, the latency and throughput benchmarks evaluate the system and provide relevant network metrics for Orfeon to optimize. The former utilizes the tcp-latency [176] project to diagnose the latency of the device, whereas the latter relies on speedtest-cli [44] to test the throughput. The metric shipper showcased here is directly related with the second contribution (C2) stated in Section 4.1.2, the metrics needed by Orfeon are obtained by the metric shipper using the tools in Table 4.2. It can be

seen that some of these metrics (e.g., gpu, performance) are very specific to the analytical domain strengthening this way the first contribution (C1). In addition, the fact that the metric shipper has been developed in Python and can be used within a container, promotes the third contribution (C3) since it can be utilized in a myriad of devices and operating systems.

4.4.3 Core

The core system contains the main functionality of the system and its overarching goal is to yield a deployment description of an analytical pipeline leveraging the predefined objectives and constraints. The high level architecture of this component is depicted in Figure 4.2, and represents the different submodules of the Orfeon (1) Optimizer. The (2) Infrastructure and the (3) Pipeline components provide the system with the capability of obtaining the mandatory inputs for the optimizer to perform its task. The former is used to obtain the infrastructure definition of the entire ecosystem, whereas the latter is able to retrieve a fully descriptive analytical pipeline in PADL language. The (4) Algorithm submodule has been designed to provide the system with the capability of utilizing different algorithms to perform its job. For this research, the implementation found in jMetalPy of the (5) NSGA-III [70] algorithm has been utilized. However, given that different problems may require the use of different approaches, further algorithms can be integrated into the system in advance using this submodule. The one used in this paper, NSGA-III, is a many-objective genetic algorithm, hence it requires the use of a (6) Mutation algorithm to function properly. As part of this research, the (7) Switch Device mutation has been specifically tailored to tackle the problem in hand and is explained in further detail in subsection 4.5.4.

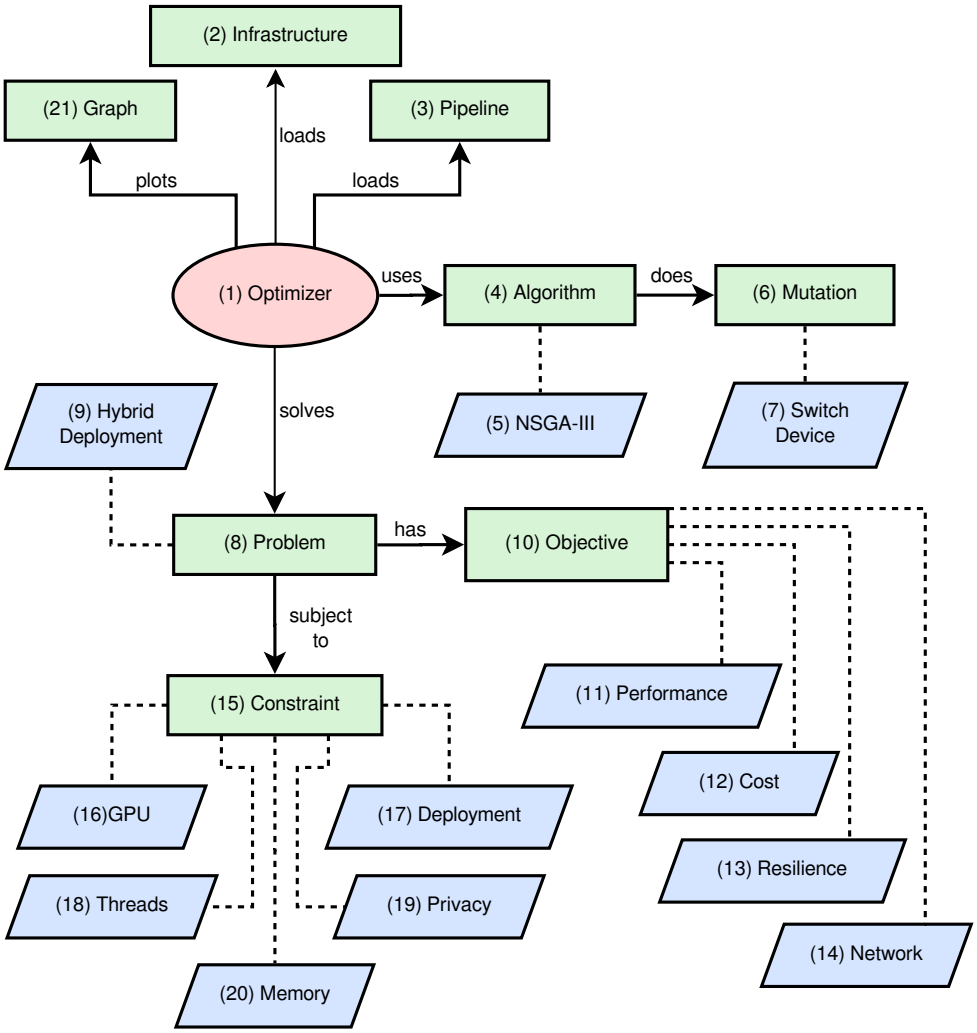

Fig. 4.2 High level architecture diagram of the core system.

Thus far, the necessary components of the optimizer have been explained. However, the main job of the optimizer is solving the problem of the deployment of distributed analytical pipelines in a predefined infrastructure. This is obtained with the (8) Problem submodule, and the (9) Hybrid Deployment, which is the formal implementation of the problem in hand and it is comprised of two main entities. Firstly, the (10) Objective component is used to address the various goals that the system needs to evaluate. Four objectives have been developed as part of this research: the (11) Performance, which represents the ability of the optimizer for utilizing the resources of the infrastructural devices in the most appropriated manner; the (12)

Cost, meaning that the optimizer will promote cheaper solutions over the most expensive ones; the (13) Resilience objective, which favours the use of better securized infrastructural devices for the different solutions; and the (14) Network objective, which pursues the network efficiency of the pipeline avoiding communication pitfalls. Secondly, the (15) Constraint submodule facilitates the use of a series of restrictions the system needs to adhere when evaluating the different solutions. In order to solve this problem, five constraints have been implemented: the (16) GPU constraint makes sure that the GPU requirements of the models do not exceed each of the infrastructural devices' capacity; the (17) Deployment constraint evaluates that every model is deployed at least in one device; the (18) Threads constraint evaluates that the CPU requirements of the pipeline are met; the (19) Privacy constraint makes sure that every model is deployed solely in locations that meet its privacy requirements, which are described in the PADL pipeline; and the (20) Memory constraint, which ensures that the memory requirements of each model are met. The Objective and Constraint submodules rely on the data retrieved by the metric shipper, and the formal implementation of them both is explained in further detail in subsection 4.5.1. Finally, the (21) Graph component provides plotting capabilities to the system, and can be used to better illustrate the results of Orfeon, hence it has been used to produce all the figures in section 4.6.

4.5 The Optimizer

This section offers an overview of the optimization process. The mathematical formalization is explained in 4.5.1. The utilized algorithm and the reasoning behind its use can be found in 4.5.2. Then, the crossover process is explained in 4.5.3. Finally, the mutation is explained in 4.5.4.

4.5.1 Problem Formulation

This subsection contains the mathematical formulation of the problem. A summary of the different variables and parameters in use by the problem can be seen in Table 4.3.

Given m models, each model $i \in 1..m$. Given d devices, each device $j \in 1..d$. Let x_{ij} be a variable that is 1 if the model i has been deployed in device j and 0 otherwise. Due to this, let the following matrix be a solution, denoted as $S \in R^{mxd}$. Let $I = 1,\ldots,m$ denote the indices of the rows of a matrix S. Let $J = 1,\ldots,d$ denote the indices of the columns of a matrix S. Let $s_{i,j}$ denote the value of the element $S[i,j]$.

Variables	
S	Binary solution matrix in which each element represents whether a model is deployed in a device or not.
a	Vector containing the thread capacity for each of the devices.
b	Vector containing the memory capacity for each of the devices.
c	Vector containing the GPU capacity for each of the devices.
e_c	Vector containing the numerical value of the country in which each device is located.
e_r	Vector containing the numerical value of the continent in which each device is located.
f	Vector containing the resiliency for each of the devices.
g	Vector containing the performance for each of the devices.
h	Vector containing the threads being executed by each of the devices.
n	Vector containing the performance penalty of each device based on the number of threads it is executing.
o	Vector containing the power consumption of each device based on the number of threads it is executing.
p	Vector containing the threads required by each of the models.
q	Vector containing the memory required by each of the models.
r	Vector containing the GPU required by each of the models.
t_c	Vector containing the numerical value of the country in which each model should preferably be placed.
t_r	Vector containing the numerical value of the continent in which each model should preferably be placed.
u_o	This vector contains a 2 for every model that should be placed in the same country it has been defined, it contains a 1 for every model that should be deployed in a device in the same continent, and a 0 if the privacy placement can be ignored.
v_s	This vector contains a 1 if the placement of the model is strict, and a 0 otherwise.
v_l	This vector contains a 1 if the placement of the model is lenient, and a 0 otherwise.
w	Vector containing the throughput for each of the devices.
z	Vector containing the latency values between the different devices.
τ	Constraint function limiting that no device is assigned to parallelize more threads than its limit.
μ	Constraint function limiting that no device is assigned to allocate more memory than its limit.
α	Constraint function limiting that each model should be deployed in at lest one device.
θ	Constraint function limiting that no device is assigned is required to have more GPU than its limit.
π	Constraint function limiting that each model is deployed based on its privacy requirements.
π_s	Function limiting that each model is deployed based on its strict privacy requirements.
π_l	Function limiting that each model is deployed based on its lenient privacy requirements.
ρ	Function evaluating the resiliency of a given solution.
ϕ	Function evaluating the model performance of a given solution.
ψ	Function evaluating the power consumption of a given solution.
ξ	Function evaluating the network performance of a given solution.

Table 4.3 Optimization model variables.

$$S = \begin{bmatrix} s_{11} & s_{12} & \dots & s_{1d} \\ s_{21} & s_{22} & \dots & s_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ s_{m1} & s_{m2} & \dots & s_{md} \end{bmatrix} \tag{4.1}$$

The system is subject to a series of constraints. The first one being that each device in the infrastructure can parallelize a certain amount of threads without incurring severe inefficiencies. Let a_j be a variable specifying this maximum capacity for the device j. In addition, let p_i be a variable holding the required threads of each model. Let x_j be a variable representing the ratio between the allocated threads and the maximum capacity for the device j.

$$\boldsymbol{x}_j = \begin{cases} \frac{\sum_{i=1}^{m} s_{i,j} p_i}{a_j} & if \sum_{i=1}^{m} s_{i,j} p_i > a_j, \\ 0 & otherwise. \end{cases}, \forall j \in J \tag{4.2}$$

Then, the CPU constraint for every device yields a zero if satisfied, and the deviation calculated in the previous step otherwise. This deviation enables Orfeon to know how far off the given solution is from fulfilling the constraint. This is formalized as follows.

$$\tau(S, \boldsymbol{p}, \boldsymbol{a}, \boldsymbol{x}) = \begin{cases} 0 & if \sum_{i=1}^{m} s_{i,j} p_i \leq a_j, \forall j \in J \\ \sum_{j=1}^{d} x_j & otherwise. \end{cases} \tag{4.3}$$

Similarly, no device can allocate more memory than its limit, failing to do so would incur in swapping, and the performance would be severely penalized. Let b_j be a variable representing the memory a device can safely allocate, and let q_i be a variable containing the required memory for each model. Aditionally, let x_j be a variable representing the ratio between the allocated memory and the maximum capacity for the device j, which is formalized as follows.

$$\boldsymbol{x}_j = \begin{cases} \frac{\sum_{i=1}^{m} s_{i,j} q_i}{b_j} & if \sum_{i=1}^{m} s_{i,j} q_i > b_j, \\ 0 & otherwise. \end{cases}, \forall j \in J \tag{4.4}$$

Then, the memory constraint yields a zero should the constraint be satisfied, and the deviation calculated in the previous step otherwise. This is formalized as follows.

$$\mu(S, \boldsymbol{q}, \boldsymbol{b}, \boldsymbol{x}) = \begin{cases} 0 & if \sum_{i=1}^{m} s_{i,j} q_i \leq b_j, \forall j \in J \\ \sum_{j=1}^{d} x_j & otherwise. \end{cases} \tag{4.5}$$

Next, yet another requirement is that for a solution to be valid, every model needs to be deployed in at least one device. This requirement represents the necessity for a given model to be operationalized in no less than one device, but with the possibility of being deployed in more (e.g., to be more resilient). In order to provide the deviation from when the constraint is not met, let x_j be a variable holding a one if model i is deployed at least once in the infrastructure, and a zero otherwise.

$$\boldsymbol{x}_i = \begin{cases} 1 & if \sum_{j=1}^{d} s_{i,j} \geq 1, \\ 0 & otherwise. \end{cases}, \forall j \in J \tag{4.6}$$

Then, the constraint yields a zero if satisfied for the given solution, and the deviation calculated in the previous step otherwise. This is formalized as follows.

$$\alpha(S, \boldsymbol{x}) = \begin{cases} 0 & if \sum_{j=1}^{d} s_{i,j} \geq 1, \forall j \in J \\ \sum_{i=1}^{m} x_i & otherwise. \end{cases} \tag{4.7}$$

In addition, artificial intelligence models often require the devices to integrate GPU processing capabilities. Accordingly, the accumulated GPU requirements of the models running on each devices should not exceed that device's capacity. Let c_j be a variable representing the GPU capacity for the device j, and let r_i be a variable containing the GPU requirements of model i. Finally, let x_j be a variable holding the deviation of device j from fulfilling this constraint.

$$\boldsymbol{x}_j = \begin{cases} \frac{\sum_{i=1}^{m} s_{i,j} r_j}{c_j} & if \sum_{i=1}^{m} s_{i,j} r_j > c_j, \\ 0 & otherwise. \end{cases}, \forall j \in J \tag{4.8}$$

Then, this constraint yields a zero if fulfiled, and the previously calculated deviation otherwise, which is formalized as follows.

$$\theta(S, \boldsymbol{r}, \boldsymbol{c}, \boldsymbol{x}) = \begin{cases} 0 & if \sum_{i=1}^{m} s_{i,j} r_i \leq c_j, \forall j \in J \\ \sum_{i=j}^{d} x_j & otherwise. \end{cases} \tag{4.9}$$

The final constraint being used in the system is one regarding the privacy of the deployed models. Each model has already been annotated with the country in which the execution will preferably be placed. Let $t_{c,i}$ be the numerical value of the country in which model i should preferably be placed. Accordingly, let $e_{c,j}$ be the numerical value of the country in which device j is located. In addition, each model comprises a privacy type representing how strict that placement should be. Let $u_{o,i}$ be a variable that takes the value 2 if the model must be

placed in the specified country, and 1 if the model should be placed in a device within the same continent. Should this variable take the value 0, this privacy constraint is ignored. First, a vector d_s is defined representing the strict placement of such models, let d_{si} be a variable that is 1 if the placement is strict (i.e. 2), and 0 otherwise. On a similar note, a vector d_l is defined representing a more lenient deployment of the models. Let d_{li} be a variable that is 1 if the placement is lenient, and 0 otherwise.

$$\boldsymbol{d}_{s,i} = \begin{cases} 1 & if \boldsymbol{d}_i = 2, \\ 0 & otherwise. \end{cases}, \forall i \in I \tag{4.10}$$

$$\boldsymbol{d}_{l,i} = \begin{cases} 1 & if \boldsymbol{d}_i = 1, \\ 0 & otherwise. \end{cases}, \forall i \in I \tag{4.11}$$

Based on these two vectors the privacy constraint is formalized as follows.

$$\boldsymbol{x}_i = \begin{cases} 0 & if s_{i,j} t_{c,i} d_{s,i} = s_{i,j} e_{c,j} d_{s,i}, \\ -1 & otherwise. \end{cases}, \forall j \in J \tag{4.12}$$

$$\pi_s(\boldsymbol{x}) = \sum_{i=1}^{m} x_i \tag{4.13}$$

$$\boldsymbol{y}_i = \begin{cases} 0 & if s_{i,j} t_{r,i} d_{l,i} = s_{i,j} e_{r,j} d_{l,i}, \\ -1 & otherwise. \end{cases}, \forall j \in J \tag{4.14}$$

$$\pi_l(\boldsymbol{y}) = \sum_{i=1}^{m} y_i \tag{4.15}$$

$$\pi = \pi_s + \pi_l \tag{4.16}$$

The optimization is performed leveraging four objectives: i) the resilience, ii) the model performance, iii) the cost, and the iv) network performance of the different solutions.

The first objective, the (i) resilience is calculated in the following manner. Let f_j be a variable holding the resiliency of the device j. The resiliency of a given model is the average security measurement of all the devices in which it has been deployed. Then, the total resiliency of a given solution is the sum of individual resiliency measurements of each individual model divided by the number of models. In order to calculate the average, we utilize the Kronecker delta formalized in Equation 4.17. This resiliency is showcased in Equation 4.18.

$$\delta_i = \begin{cases} 1 & if i = 0, \\ 0 & otherwise. \end{cases} \tag{4.17}$$

$$\rho(S, \boldsymbol{f}) = \frac{1}{m} \sum_{i=1}^{m} \frac{\sum_{j=1}^{d} s_{i,j} f_j}{\sum_{j=1}^{d} 1 - \delta_{s_{i,j}}} \tag{4.18}$$

The second objective, the (ii) model performance, is calculated considering that devices perform better the less effort they need to undergo in order to parallelize different jobs. However, the parallelization being made by the different devices comes at a cost, and we represent this by adding a penalty accounting for the number of threads each device is executing. Let p_i be a variable which contains the number of threads needed to execute model i, and let x_j be the number of threads being executed by device j. This is formalized as follows.

$$x_j = \sum_{i=1}^{m} s_{i,j} p_i \tag{4.19}$$

To this end, let g_j be a variable which contains the performance of device j. Then, let n_{jh_j} be a variable containing the performance penalty of device j, when executing h_j threads. This way, the performance of each device can be adjusted with a correction coefficient in order to get a more realistic value. Let matrix E be the matrix containing the performance adjusted. It has been established in this research that the performance of a model is the average performance obtained out of all the devices in which it is to be deployed. Then, the performance value of the solution is the sum of the individual performance of each model divided by the number of models. This is formalized as follows.

$$\phi(S, \boldsymbol{g}, \boldsymbol{n}, \boldsymbol{x}) = \frac{1}{m} \sum_{i=1}^{m} \frac{\sum_{j=1}^{d} s_{i,j} g_j n_{j,x_j}}{\sum_{j=1}^{d} 1 - \delta_{s_{i,j}}} \tag{4.20}$$

The (iii) cost goal seeks for the solutions to be as cheap and environmentally friendly as possible. Summarizing, once the Hyper-Threading feature kicks in, and parallelization starts taking place in virtual cores, power consumption does not increase linearly, as all the physical cores are already in use. Let q_j be a variable containing the number of threads in use by device j. Then, let o_{jk} be a variable containing the power consumption of device j, when executing k threads. This constraint is formalized as follows.

$$\psi(S, \boldsymbol{o}, \boldsymbol{q}) = \frac{1}{d} \sum_{i=1}^{m} \sum_{j=1}^{d} s_{i,j} o_{j,q_j} \tag{4.21}$$

Finally, the (iv) network performance fitness is calculated leveraging two metrics gathered by the metric shipper: the throughput, and the latency. This fitness is devoted to minimize the distance between the pipeline stages that communicate with each other, and favours devices with higher throughput. Let $z_{j,k}$ be a variable containing the communication latencies between devices j and k. Then, let $x_{i,j,k}$ a variable containing the communication latency of model i, which is deployed in device j with device k.

$$x_{i,j,k} = s_{ij} z_{jk}, \forall i \in I, \forall j \in J, \forall k \in J \tag{4.22}$$

Then, we have defined the network performance fitness of a solution as the substraction of the throughput and the latency of the given solution. Let w_j be a variable which contains the throughput for device j. Firstly, the throughput of the solution is the sum of the average throughput of the different models. Secondly, let $y_{i,k}$ be a variable which states wether model i needs to communicate with device k. Then, the total latency of the solution is the sum of the communication latencies of the different models between them. The network performance fitness is then defined as the substraction of these two items, which is formalized as follows.

$$\xi(S, \boldsymbol{e}, \boldsymbol{x}, \boldsymbol{y}) = \frac{1}{m} \sum_{i=1}^{m} \frac{\sum_{j=1}^{d} s_{i,j} e_j}{\sum_{j=1}^{d} 1 - \delta_{s_{i,j}}} - \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{d} \sum_{k=1}^{d} x_{i,j,k} y_{i,k} \tag{4.23}$$

Summarizing, the entire problem can be defined as the minimization of the cost formalized in Equation 4.26, and the maximization of the performance shown in Equation 4.25, and the resiliency which is formalized in Equation 4.24, all of them subject to the series of constraints described above. All of this is formalized in the following way.

$$\max \frac{1}{m} \sum_{i=1}^{m} \frac{\sum_{j=1}^{d} s_{i,j} f_j}{\sum_{j=1}^{d} 1 - \delta_{s_{i,j}}} \tag{4.24}$$

$$\max \frac{1}{m} \sum_{i=1}^{m} \frac{\sum_{j=1}^{d} s_{i,j} g_j n_{j,x_j}}{\sum_{j=1}^{d} 1 - \delta_{s_{i,j}}} \tag{4.25}$$

$$\min \frac{1}{d} \sum_{i=1}^{m} \sum_{j=1}^{d} s_{i,j} o_{j,q_j} \tag{4.26}$$

$$\max \left(\frac{1}{m} \sum_{i=1}^{m} \frac{\sum_{j=1}^{d} s_{i,j} e_j}{\sum_{j=1}^{d} 1 - \delta_{s_{i,j}}} - \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{d} \sum_{k=1}^{d} x_{i,j,k} y_{ik} \right) \tag{4.27}$$

Subject to:

$$\sum_{i=1}^{m} s_{i,j} p_i \leq a_j, \forall j \in J \tag{4.28}$$

$$\sum_{i=1}^{m} s_{i,j} q_i \leq b_j, \forall j \in J \tag{4.29}$$

$$\sum_{j=1}^{d} s_{i,j} \geq 1, \forall j \in J \tag{4.30}$$

$$\sum_{i=1}^{m} s_{i,j} r_i \leq c_j, \forall j \in J \tag{4.31}$$

$$s_{i,j} t_{c,i} d_{s,i} = s_{i,j} e_{c,j} d_{s,i}, \forall i \in I, \forall j \in J \tag{4.32}$$

$$s_{i,j} t_{r,i} d_{l,i} = s_{i,j} e_{r,j} d_{l,i}, \forall i \in I, \forall j \in J \tag{4.33}$$

This section directly address the fourth contribution (C4) stated in Section 4.1.2, as we have proposed the mathematical formulation for the goal-driven operationalization of distributed analytical pipelines.

4.5.2 The algorithm

In order to solve this scenario in which several objective functions need to be optimized simultaneously subject to a set of constraints, we resort to the area of multi-objective optimization (MOO). These algorithms have been successfully applied in many fields such as engineering [49], chemistry [251], and logistics [41]. It is worth mentioning that the overarching goal of MOO algorithms is to find a group of nondominated solutions that balance a set of conflicting objectives balancing in a Pareto optimal front. One of the most utilized MOO algorithms is the nondominated sorting genetic algorithm II [69, 71] (NSGA-II), which was conceived to alleviate various difficulties of multi-objective evolutionary algorithms: computational complexity, nonelitism approach, and the need for specifying a sharing parameter. In spite of its age, NSGA-II is actively utilized in research nowadays [293, 312, 302], and it is widely accepted as a reliable algorithm for multi-objective optimization. An evolution of this algorithm is presented in [70, 148], in which the authors propose NSGA-III, a many-objective optimization algorithm that emphasizes population members that are not dominated. Initially, a random population is generated. Then, the population is sorted and a binary tournament selection, crossover and mutation is applied to the initial population to generate the offspring. In order to ensure elitism, both the offspring and the initial population

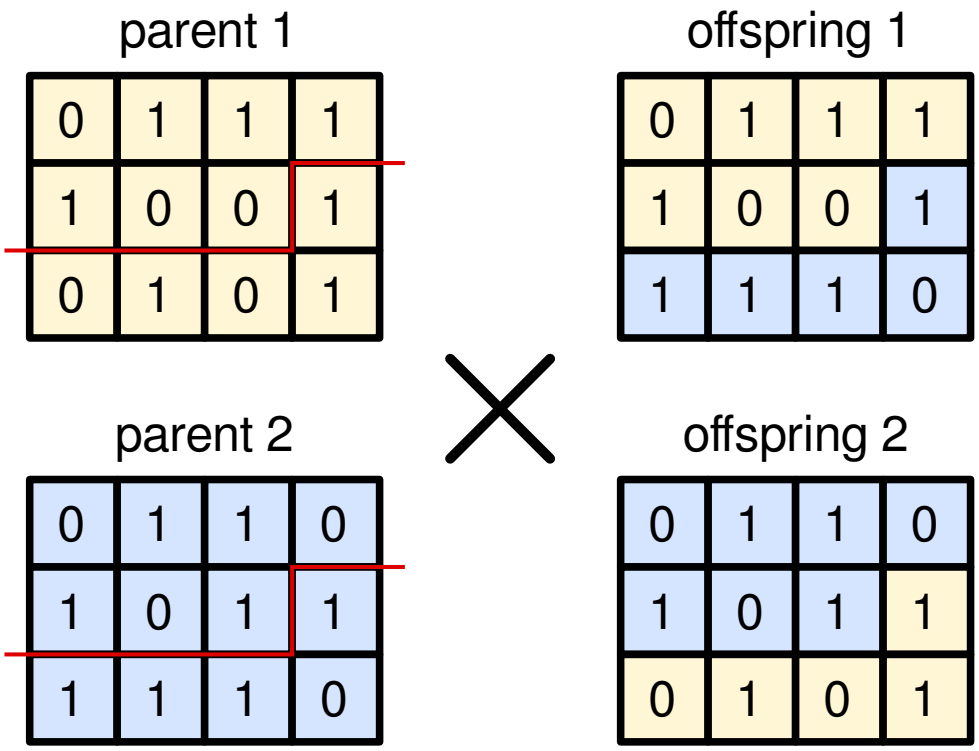

Fig. 4.3 Crossover operator between two individuals of the population.

are sorted. Finally, a new population is generated with the best individuals of the offspring and the initial populations.

4.5.3 Crossover

The crossover operator refers to the process of two individuals generating offspring based on the genetic material of their parents. The crossover is of paramount importance for the convergence of the algorithm. In this paper, we have utilized the single-point crossover as depicted in Figure 4.3. In this type of crossover both parents are cut at a random location. Then, the genetic information of the first parent up to that point combines with the genetic information of the second parents from that point onwards to produce the first child. Conversely, the second child holds the genetic information of the second parent until the crossover point combined with the genetic information of the first parent from that point onwards.

4.5.4 Mutation

In this scenario, the solution variable represents whether an analytical model is to be deployed in a particular device. In this research, three different mutation tackling different goals have been utilized: the i) Free Up mutation, the ii) Bit Flip mutation, and the iii) Deployment mutation.

The first mutation, the free Up mutation, can be seen in Algorithm 1. For every device in the solution variable S, there is a probability denoted by variable $probability$ of kicking off a free up task in the selected device i. The algorithm evaluates every model in the solution variable, and has a 50% probability of performing and undeployment of model j in device i. This mutation promotes the convergence of the cost objective.

Algorithm 1: Free Up Mutation.

```
Input: S, probability, models, devices
for i ← 1 to devices do
    rand ← random();
    if rand ≤ probability then
        for j ← 1 to models do
            rand ← random();
            if rand ≤ 0.5 then
                s_{j,i} ← False;
            end
        end
    end
end
```

The second mutation, the Bit Flip mutation, can be seen in Algorithm 2. Firstly, for every value in the solution variable S there is a possibility denoted by variable $rand$ of switching that value to the opposite (i.e. if the evaluated bit contains a 1, it will be converted to a 0, and vice versa). This represents the possibility of a model of switching its status from undeployed to deployed or viceversa.

Finally, the Deployment mutation is showcased in Algorithm 3. This mutation iterates over all the models in the solution variable S and sum the number of times that model has been deployed. Should the model be undeployed, it iterates over the different infrastructural devices, and forces a deployment based on a probability denoted by $probability$. This mutation is convenient because any undeployed model leads to an invalid solution, hence it promotes the fulfilment of the deployment constraint.

Algorithm 2: Bit Flip Mutation.

```
Input: S, probability, models, devices
for i ← 1 to models do
    for j ← 1 to devices do
        rand ← random();
        if rand ≤ probability then
            if s_{i,j} = True then
                s_{i,j} ← False;
            else
                s_{i,j} ← True;
            end
        end
    end
end
```

4.6 Experimental Results

This section presents the main results of the process undergone to validate the proposed Orfeon system. The following resources are presented: (4.6.1) the setup to implement the different experiments, (4.6.2) the analysis of the results from a scalability perspective, (4.6.3) the fulfillment of the different objectives that direct the optimization performed by the Orfeon system, (4.8) a discussion of the proposed framework.

4.6.1 Experimental Setup

In order to perform the evaluation of the Orfeon solution under realistic working conditions, we have envisioned a scenario in which a company is willing to utilize a distributed infrastructure to run their analytical workloads. We showcase how Orfeon is a suitable tool to be used in this scenario, to take advantage of the deployment of the small analytical pipelines as well as the large and complex ones. In particular, it can be beneficial for leveraging the benefits of the different computational layers that are available to companies (e.g, edge, on-premises, cloud), and to ensure that the privacy concerns of such companies are met, among others.

This scenario, which is shown in Table 4.4, corresponds to a company located in Spain which has traditionally executed the bulk of their computation on-premises. Initially, they acquired six (iii) Dell Precision 3520 workstations in which they were able to perform their whole computation. In addition, they have sixteen (i) edge devices that were initially utilized for gathering the data. However, they could not harness their computational power and were

Algorithm 3: Deployment Mutation.

```
Input: S, probability, models, devices
for i ← 1 to models do
    x ← { 1  if s_{i,j} = 0, ∀i ∈ I, ∀j ∈ J
        { 0  otherwise.                    ;
    if x = 0 then
        for k ← 1 to devices do
            rand ← random();
            if rand ≤ probability then
                s_{i,k} ← True;
            end
        end
    end
end
```

severely underutilized, this is a scenario in which Orfeon is extremely beneficial since it will be able to select the most appropriate analytical models to be run on these edge devices. Eventually, the requirements of the company increased and, given that they already owned a server room and qualified professionals, they acquired ten ii) PowerEdge R440 servers to be utilized in their facilities. At some point, the workloads that they needed to run on their computational devices became too large, and they evaluated the possibility of expanding their infrastructure. However, after many considerations they decided to harness the computational power being offered by cloud providers, in particular the AWS stack. They decided to use eight (v) r5a.8xlarge on-demand instances located in the United States, which offers lower costs than their European counterparts. This architecture was sufficient for running most of their analytical pipelines, but the company identified that some workloads were only suited for being executed in Europe due to the differences in the legislation. For this reason, they decided to utilize eight (iv) t3.2xlarge instances located in Europe, they can constraint certain workloads to only be run in Europe by using the privacy annotations PADL offers for this matter. Finally, in order to increase the execution speed of some important workloads, the company decided to utilize two (vi) g3s.xlarge on-demand instances, with a built-in GPU to speed up the most heavyweight analytical pipelines.

The implementation of this scenario has been done in the following manner. The (i) edge devices have been virtualized using Oracle VM VirtualBox [217] and Vagrant [123], they represent humble infrastructural devices used in the field to gather the necessary information used within the analytical pipelines. They are appropriate for running lightweight analytical models due to their low consumption, and latency. For the devices that can be found (ii,

Type	Number of Devices	Layer	Location	Clock Speed (GHz)	vCPUs	Memory (GiB)	GPUs	GPU Memory (GiB)
(i) VirtualBox	16	Edge	ES	3.00	2	6	-	-
(ii) PowerEdge R440	10	Premises	ES	1.80	32	251	-	-
(iii) Precision 3520	6	Premises	ES	4.0	8	15	-	-
(iv) t3.2xlarge	8	Cloud	IE	2.4	8	31	-	-
(v) r5a.8xlarge	8	Cloud	US	2.5	32	256	-	-
(vi) g3s.xlarge	2	Cloud	US	3.0	4	30.5	1	8

Table 4.4 Experimental Scenario of an Infrastructure Spanning throuhout Edge and Cloud.

iii) on-premises, bare metal servers available have been used. Finally, the (iv, v, vi) cloud instances are generated on demand using the AWS stack. The provisioning of these machines has been done with Terraform [122], and the configuration management tool selected for installing and configuring the dependencies is Ansible [124]. Once the infrastructure defined in Table 4.4 has been provisioned and configured, the metric shipper has been deployed in each of the infrastructural devices and the metrics have been centralized in a cloud database. These metrics conform to the infrastructure definition required by the Orfeon optimizer to leverage the aforementioned objectives and constraints, and have been utilized across Section 4.6 for all the experiments. In addition, this subsection serves as validation for the metric shipper, as it has been able to appropriately query, benchmark, and collect the heterogeneous distributed infrastructure described in this section. Finally, the fact that the metric shipper is able to utilize all the devices in Table 4.4, which span across the edge and cloud layers, promotes contribution number three (C3). In addition, the fifth contribution (C5) is strengthen since Orfeon has been able to utilize several Infrastructure as Code (IaC) technologies (e.g., Terraform, Ansible, Vagrant) and cloud providers (e.g., AWS).

4.6.2 Scalability Analysis

This subsection offers an evaluation of Orfeon under varying conditions in order to assess its suitability for different workloads, ranging from very small to exceptionally large. This evaluation has been performed in three infrastructural devices described in Table 4.4: a Dell Precision 3520, a PowerEdge R440, and t3.2xlarge. In it, we try showcase how Orfeon is able to maintain appropriate execution times and memory footprint under the different scenarios presented.

Figure 4.4 offers a graphical description of the different execution times of the Orfeon platform. It is worth pointing out that we are benchmarking Orfeon's core, which considers the entire infrastructure but takes place in a single machine. This analysis has been performed in three of the infrastructural devices to asses if the results are consistent across them. On the one hand, the vertical axis represents the average execution times of a hundred runs of the

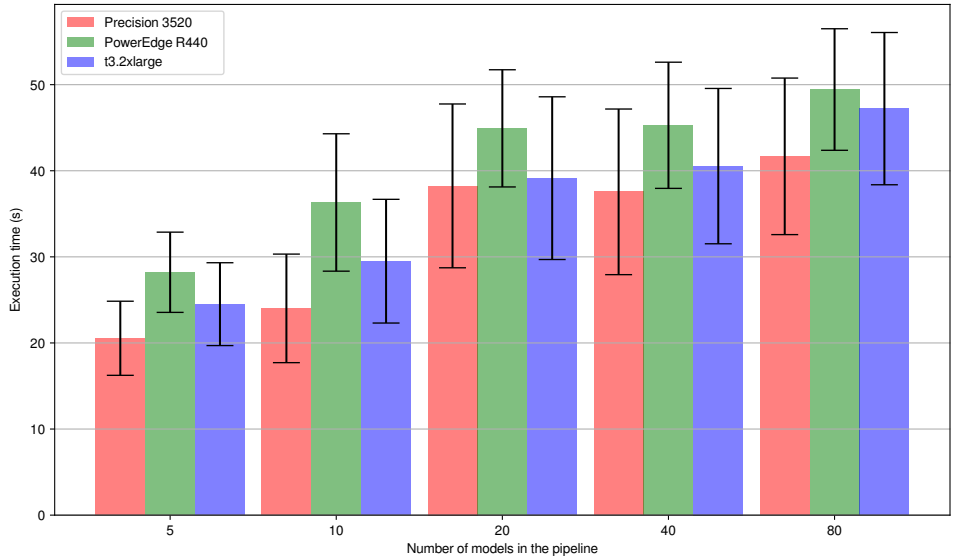

Fig. 4.4 Average execution times of one hundred runs of the optimizer with varying pipeline sizes in different platforms.

platform, with the appropriate error bars. On the other hand, the horizontal axis represents the number of models for which their deployment is optimized using the infrastructure shown in Table 4.4. It is worth noting that the number of models evaluated is duplicated for every new scenario. In addition, given that every model can be deployed in more than one device (e.g., due to high availability, redundancy), the number of possibilities for each new scenario grow exponentially. For instance, the number of available scenarios for five models, in a fifty infrastructural devices scenario is $2^{50 \cdot 5}$ which is $1.81 \cdot 10^{75}$. In the ten model scenario this number goes up to $3.27 \cdot 10^{150}$, and rises to $1.07 \cdot 10^{301}$ in the twenty model scenario. In spite of these numbers being daunting to fathom, Orfeon is able to maintain an steady increase in the execution times even in the eighty models scenario, with over $1.31 \cdot 10^{1204}$ possibilities.

On the other hand, Figure 4.5 offers the memory consumption for different executions of Orfeon. We have evaluated the consumption with varying number of models, and profiled the evolution of the memory consumption over time. In order to obtain the best possible measurements we have executed Orfeon using a popular heap profiler, Massif [279]. This tool is able to execute the program on an encapsulated environment and provide accurate information about the memory consumption (e.g. heap blocks, stack sizes). As it can be seen in the figure, Orfeon was able to maintain a steady use of memory during the whole execution for all the different workloads. In addition, all the workloads utilized in this simulation were

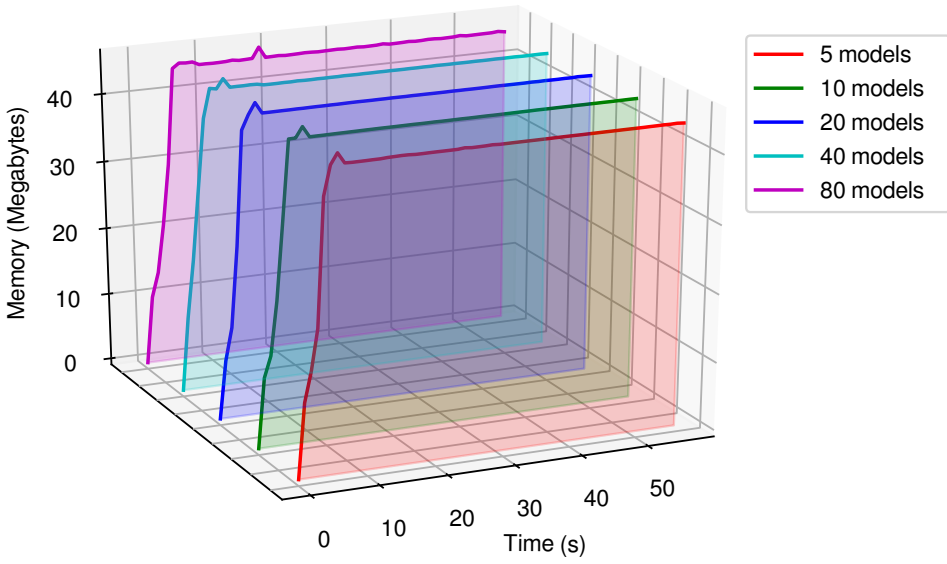

Fig. 4.5 Memory consumed by Orfeon with varying pipeline sizes.

able to use a remarkably similar amount of memory, ranging from 41.42MB, the smallest, to 41.83MB the biggest. In conclusion, Orfeon shows an appropriate behavior with different workloads, by being able to utilize a constant use of memory over time.

4.6.3 Objective Analysis

In this section we evaluate the individual evolution of the different objectives in Orfeon after a two thousand generations execution. For this analysis we have evaluated the deployment of a twenty model pipeline in the infrastructure described in Table 4.4, and Orfeon performs the optimization against all four objectives. The horizontal axis represents the number of generations, whereas the vertical axis showcases the values of the top five solutions for the objective under study. Note that even though the solutions take values between zero and one for the different objectives, this does not mean that those thresholds are attainable, as the constraints that are imposed on the system make them unreachable. It rather serves as a convinient comparison framework for different pipelines, infrastructural devices, and objectives.

Figure 4.6 shows the mean evolution of the top five solutions for the resilience objective. It can be seen that this metrick asymptotically approaches its theoretical limit, slightly

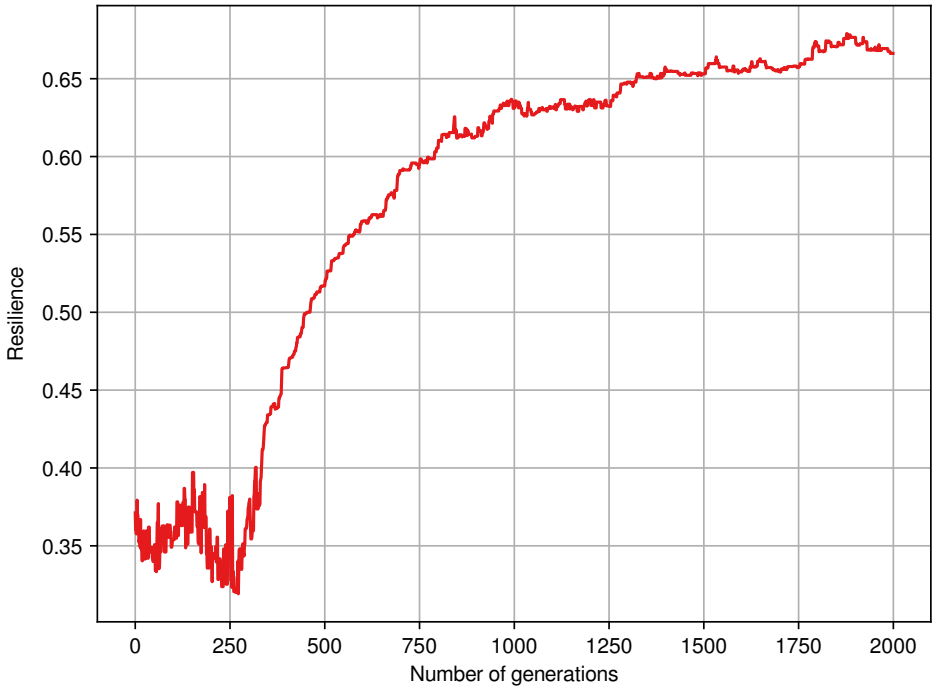

Fig. 4.6 Evolution of the top five solutions for the resilience objective over two thousand generations.

hovering over the 0.65 value. This metric is optimized based on the output given by the Lynis tool, which leverages a large variety of configurations within the managed servers (e.g., ssh, sysctl, upgrades policy, default umask, amongst others).

Similary, Figure 4.7 plots the improvement of the model performance metric over the generations, in which Orfeon is able to quickly find solutions that have a higher performance. In this objective, the metric shipper retrieves the performance of the infrastructural devices using the AI-Benchmark tool. In addition, Orfeon is able to leverage the number of threads in use by each device to yield a more realistic value.

Thirdly, Figure 4.8 showcases the evolution of the cost metric, in which Orfeon constantly favors solutions with less overall cost. It can be seen that the fitness function is appropriate for decreasing the overall cost of the solution space from a very inefficient starting point. This metric is initially collected in different ways, public cloud providers offer the hourly cost of their instances directly. However, for bare metal solutions the consumption is gathered

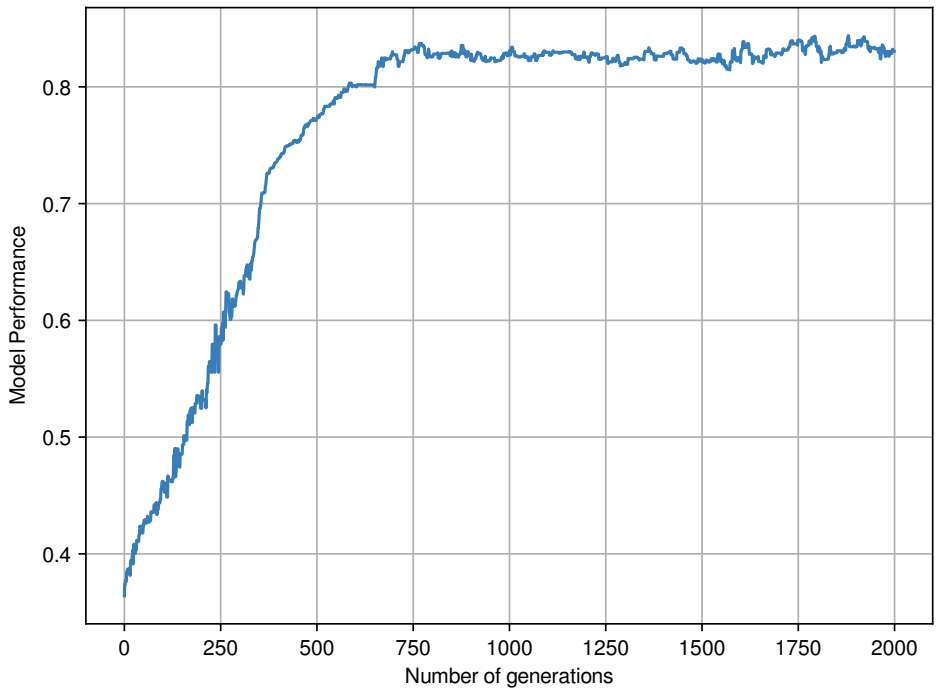

Fig. 4.7 Evolution of the top five solutions for the model performance objective over two thousand generations.

either programmatically or with a power meter when necessary, and leverages the CPU, GPU, and memory load of the infrastructural devices under different scenarios.

Lastly, Figure 4.9 represents the evolution of the network performance metric. Initially this fitness function starts from a very low value, which represents the sparcity of the deployed analytical pipeline, and asymptotically approaches its limit. Initially, the values for this fitness are particularly noisy, and it is not until the three hundred generation they stabilize. This is because Orfeon starts favoring the fulfilment of the different constraints imposed on the system, and only then the values of the different fitness start to imporve. For this objective, Orfeon leverages the throughput and latency amongst the different infrastructural devices. These metrics are collected by the metric shipper utilizing the speedtest-cli and tcp-latency libraries, respectively.

Finally, Figure 4.10 showcases the solutions for a forty model analytical pipeline obtained with a two hundred individuals population size. The vertical and horizontal axes represent the different fitnesses (i.e, resilience, model performance, cost, network performance). The

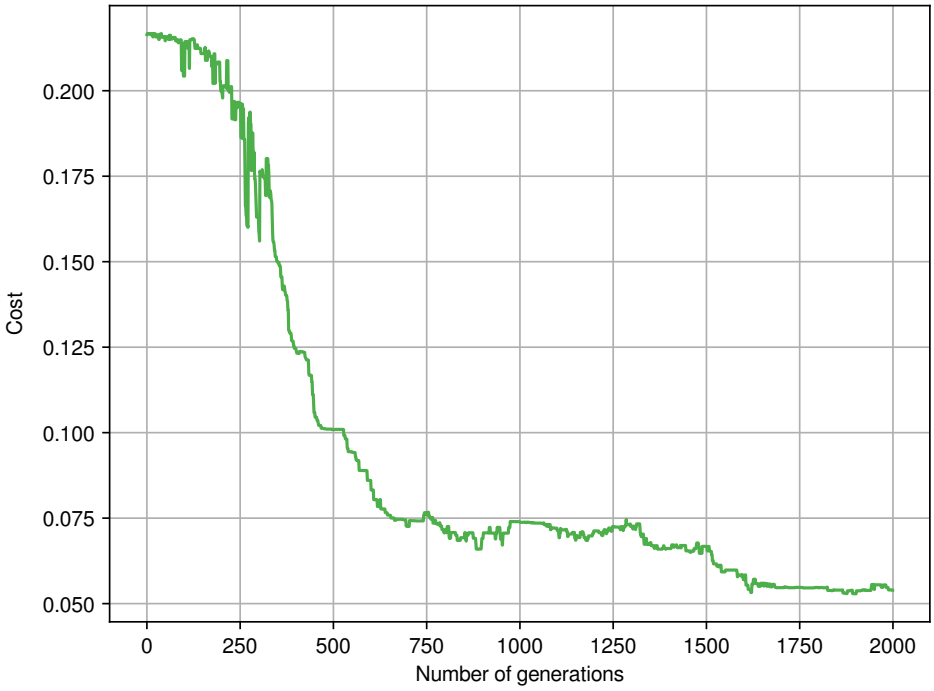

Fig. 4.8 Evolution of the top five solutions for the cost objective over two thousand generations.

comparison is made by plotting the two-dimensional representation of each solution against each other. For each subplot, the colored points are highlighted since they represent the pareto front obtained for the given objectives, whereas the grayed out points are solutions that are dominated by some others for those particular objectives. For the resilience and model performance objectives, it can be seen that the solutions are trending to the upper right corner, but none has been found that maximises both. Next, decreasing the cost of the solutions inevitably leads to lower values for the resilience, model performance, and network performance fitnesses. For the resilience and model performance, an increase in the latter leads to a mild decrease in the former, up to a point in which the resilience decreases drastically off a cliff. Finally, increasing the network performance steadily decreases the model performance. In summary, the cost objective is radically opposed to each and everyone of the different objectives, as trying to minize the cost inevitably leads to lower values of the rest. This is not the case for the resilience and network performance as they seem to coexist

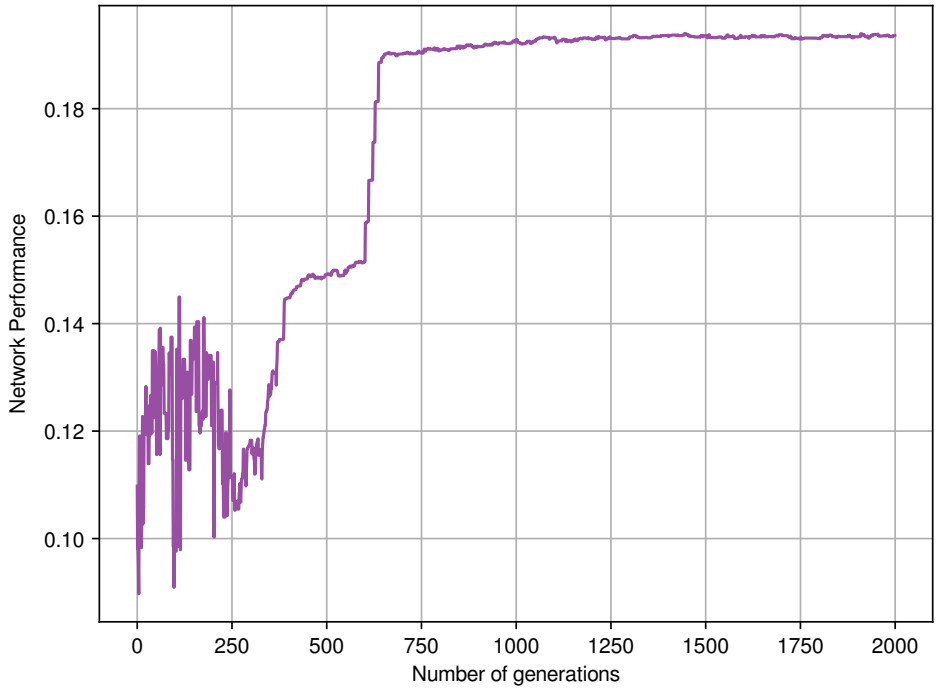

Fig. 4.9 Evolution of the top five solutions for the network performance objective over two thousand generations.

seamlessly, as an increased security on the infrastructural devices do not necessarily interfere with the networking.

4.7 Expert Evaluation

In this section, four experts versed in the operationalization of analytical pipelines onto hybrid architectures have completed different deployment scenarios leveraging the resilience, performance, cost, and networking of the provided solutions. The participants considered a forty model analytical pipeline encompassing network, privacy, and hardware constraints; and the fifty device infrastructure described in Table 4.4.

The research activities are framed under the following scenario. “Company A, which devotes itself to the orchestration of analytical pipelines, requires the expertise of a solution architect for the optimal deployment of a project on their infrastructure. The infrastructural devices span accross the cloud, edge, and on-premise layers. The goal is to provide a solution

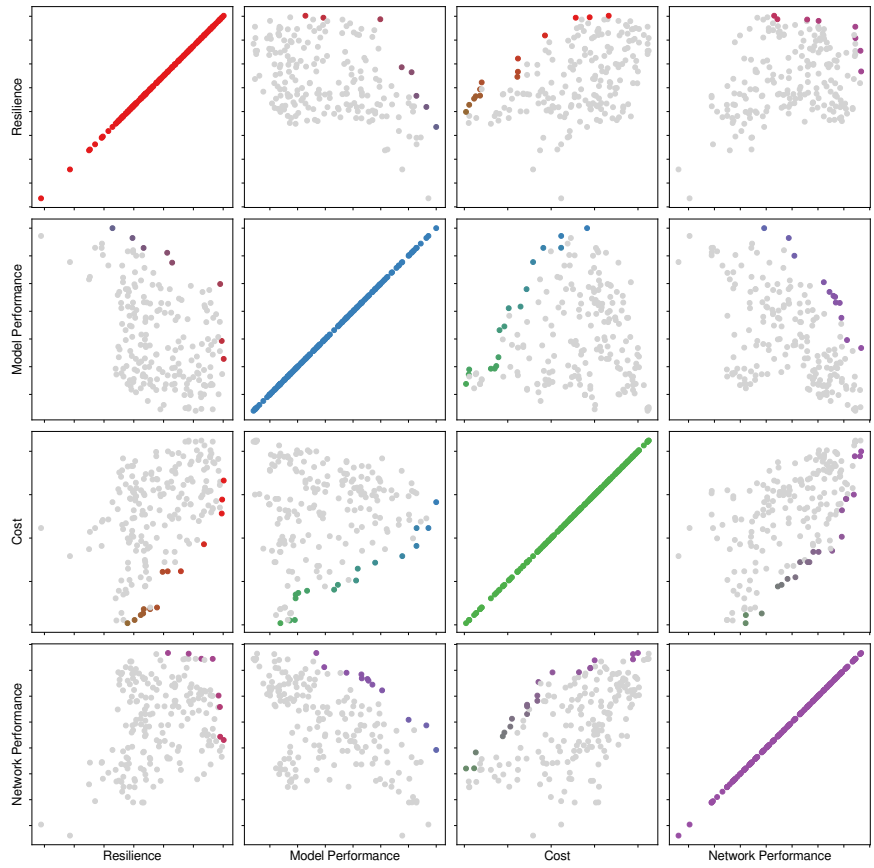

Fig. 4.10 Comparison of the different objectives obtained by using a two hundred individuals population with a forty model analytical pipeline.

for each of the different goals, but without neglecting the rest.". The experts have been asked to comply with the following statements:

- Access to the different infrastructural devices is granted in order to better understand the systems.
- Publicly available resources such as web pages can be checked to get further information.
- The solutions must comply with the different constraints imposed on the system.
- Provide a solution for each of the objectives: resilience, model performance, cost, and network performance.

- Utilize no more than two hours in tailoring all the different scenarios.

The study is structured in the following manner. A panel has been organized with the experts, in which they have been been given a summary of the different elements they have at hand (i.e., analytical pipeline, infrastructural devices, online resources), and they have been instructed to provide their solutions to be compared with the Orfeon framework. This panel has been lead by an author (A.A.) with experience in panel moderation.

	Goal	CPU	RAM	GPU	Deploy	Privacy	Value
Orfeon	R[1]	✓	✓	✓	✓	✓	0.7125
Expert 1	R[1]	✓	✓	✓	✗	✓	0.2695
Expert 2	R[1]	✗	✓	✓	✓	✗	0.3352
Expert 3	R[1]	✓	✗	✓	✓	✗	0.3254
Expert 4	R[1]	✗	✓	✓	✓	✗	0.5344
Orfeon	MP[2]	✓	✓	✓	✓	✓	0.8272
Expert 1	MP[2]	✓	✓	✓	✓	✗	0.5552
Expert 2	MP[2]	✗	✓	✓	✓	✗	0.6196
Expert 3	MP[2]	✓	✓	✓	✓	✓	0.6636
Expert 4	MP[2]	✓	✓	✓	✓	✗	0.6105
Orfeon	C[3]	✓	✓	✓	✓	✓	0.0282
Expert 1	C[3]	✓	✓	✓	✓	✓	0.0884
Expert 2	C[3]	✗	✗	✓	✓	✓	0.0674
Expert 3	C[3]	✓	✓	✓	✓	✗	0.0512
Expert 4	C[3]	✓	✓	✓	✓	✗	0.1348
Orfeon	NP[4]	✓	✓	✓	✓	✓	0.2092
Expert 1	NP[4]	✓	✓	✓	✓	✓	0.1084
Expert 2	NP[4]	✗	✓	✓	✓	✓	0.1546
Expert 3	NP[4]	✓	✓	✓	✓	✗	0.0977
Expert 4	NP[4]	✗	✓	✓	✓	✗	0.1464

[1] Resilience [2] Model Performance [3] Cost [4] Network Performance

Table 4.5 Comparison of the experts evaluation and Orfeon deploying a forty model pipeline into fifty infrastructural devices.

The results of the experts evaluation can be seen in Table 4.5. The grayed out lines correspond to Orfeon followed by those of the experts. Interestingly, being able to fulfill the different constraints have been a huge drawback for the experts. The cpu constraint, which states that no infrastructural device should execute more threads than its limit, has been met by 62.5% of the solutions. Even though some deployments failing to satisfy this constraint could theoretically work, their actual performance would be severily penalized, as the devices

would be forced to prioritize several processes at the same time. Secondly, the memory constraint, which represents that no device should allocate more memory than its limit has been met by 87.5% of the solutions. This is a more stringent constraint than the previous one, overallocating RAM memory in an infrastructural device means having to resort to swap memory (if present), which is generally orders of magnitude slower than its counterpart. This means the pipeline will fail to work properly on such deployments. Thirdly, the experts have been able to always comply with GPU requirements. There are two infrastructural devices with GPUs, which has significantly simplified this task. Next, the deployment constraint has been satisfied in 93.75% of the solutions. Failing to comply with this constraint would lead to a malfunctioning deployment, as one step of the pipeline would simply not be present. Finally, merely 37.5% of solutions provided by the experts were able to comply with the privacy constraint. This is not intended to be a functional constraint, meaning that failing to satisfy it still yields a working pipeline. Instead, it is inteneded to serve as a manner to comply with stringent privacy requirements imposed by stakeholders (e.g., sensitive data that can only be utilized in a certain country). Overall, just 18.75% of the solutions provided by the experts are considered valid and comply with all the constraints. In regards to the different objectives, not only has Orfeon complied with all the constraints, but it has also produced solutions that improve those of the experts for every single objective under analysis. In addition, Orfeon has been able to do this while optimizing all four of the objectives at the same time.

In summary, the process of orchestrating a service deployment over an infrastructure is a tedious and error prone process. Orfeon is able to automatize this procedure and yield near optimal solutions that comply with the different constraints imposed on the system by the operations (e.g., cpu, ram), development (e.g., deploy, network), and business (e.g., privacy, cost) units of an organization.

4.8 Discussion

In this research, we highlight the complexity of the machine learning lifecycle and how it differs from the traditional software development lifecycle. Due to this, we offer an MLOps and AIOps framework that aids in the operationalization of analytical distributed pipelines, and aspires to raise the adoptions of such projects in production environments. There have been a series of challenges that have required our special attention. Firstly, the metric shipper explained in Section 4.4.2 operates on the different layers showcased in Table 4.4, hence it has to overcome the connectivity issues of these heterogeneous environments. For this reason; the centralized database has been deployed in the cloud so it can be reached by all of the infrastructural devices; and password and address filtering have been utilized to

secure the environment. Then, we have found no information about the performance of infrastructural devices while executing machine learning workloads, and it is hard to infer this from the hardware capabilities (i.e. threads, RAM, GPU) of a system. Due to this, we devote to an AI benchmark that is executed by the metric shipper for obtaining this information. However, the large heterogeneity of the infrastructural devices makes this execution cumbersome, hence we rely on container-based solutions in order to promote the portability of the solution. In addition, being able to optimize four objectives at the same time has proven to be a cumbersome process. In order to promote the convergence of the different solutions, both the constraints and the fitness functions provide Orfeon with high granularity. This enables Orfeon to rank compliant and not compliant solutions appropriately. Finally, public cloud providers provide detailed pricing schemas for the use of their services, but obtaining this metric for edge and on-premises devices is not so straightforward. For the former we have utilized a power meter to accurately measure the consumption of the devices under different workloads, for the latter the approach has been threefold: i) large servers often provide dedicated tools that give this information, ii) obtaining it via software as explained in Table 4.2, and iii) utilizing the power meter when physical access to the devices was feasible.

Next, in order to asses the feasibility of Orfeon as a tool for the optimal deployment of analytical pipelines in distributed heterogeneous environments, we offer an evaluation based on the criteria showcased in Table 4.1.

F1 Distributed Infrastructure: Orfeon has been able to gather the necessary information using the metric shipper described in section 4.4.2, in the infrastructure spanning across the edge, and cloud (both private and public) computing layers showcased in Table 4.4.

F2 Machine Learning: the metric shipper (Section 4.4.2) offered alongside Orfeon provides metrics (e.g., GPU, Performance) that are very specific to the machine learning domain. In addition, using PADL description language, Orfeon is able to provide insights about the models that require deployment. Due to this, Orfeon becomes appropriate for the operationalization of analytical pipelines.

F3 Analytical Pipelines: as stated in Section 4.4, Orfeon is able to fathom the nuances of analytical pipelines using PADL description language, and provides solutions that comply with these dependencies and constraints. Accordingly, the experiments in Section 4.6.2, and Section 4.6.3 have been performed with PADL defined analytical pipelines of different sizes.

F4 Scalability: the metric shipper is able to gather the necessary information from the heterogeneous distributed infrastructure as explained in section 4.6.1. In addition,

section 4.6.2 offers an in depth evaluation of the scalability requirements of Orfeon both in terms of time and memory consumption.

F5 Goal-Driven Deployment: section 4.6.3 showcases that Orfeon is a suitable tool for the optimization of goal driven deployments in distributed analytical environments. It is able to provide a Pareto front leveraging the four evaluated objectives (i.e. resilience, performance, cost, network) and constraints (i.e. GPU, Deployment, Threads, Privacy, Memory).

F6 Open License: Orfeon has been released under the Apache License, and both the metric shipper and the core system are publicly available in GitHub [74].

F7 Technology Agnostic: Orfeon does not try to solve the various problems that can arise in the operationalization of artificial intelligence projects. Hence, it can coexist with other tools more suitable for each of the different tasks that comprise the machine learning ecosystem. For instance, PADL is used for the description of the analytical pipelines, whereas the packaging can be solved with Docker [82].

The contributions of Orfeon and this research as a whole have been presented earlier in Section 4.1.2. In addition to pointing out throughout the manuscript, we summarize in the following list how this contributions have been fulfilled by Orfeon.

C1 Orfeon has been specifically tailored to be specific to the analytical domain. It utilizes PADL which is regarded as a description language for analytical pipelines. In addition, some of the key metrics of the metric shipper are very focused on analytics (Table 4.2). Furthermore, it can be seen that Orfeon is able to solve the problem of the operationalization of analytical pipelines without incurring on scalability issues in terms of time (Figure 4.4), and memory (Figure 4.5); while being able to optimize the four objectives succesfully (Figure 4.6, Figure 4.7, Figure 4.8).

C2 The details of the metric shipper offered as part of this research are in Section 4.4.2, and the source code available to be used in GitHub [74].

C3 The Orfeon framework is envisioned to be cross-platform. Due to this, it has been implemented in Python and it can be shipped as a container in order to promote its portability (Section 4.4.2). In addition, it has been sucesfully utilized to gather metrics from heterogeneous infrastructural devices spanning accross the edge, and cloud computation layers (Table 4.4).

C4 The mathematical formulation of the system for the goal-driven deployment of analytical pipelines is explained in Section 4.5.1.

C5 Orfeon does not aspire to replace other established solutions already in use in the field. In Section 4.4 we build uppon PADL for the description of the analytical pipelines, and in Section 4.6.1 IaC technologies are used for setting up the scenario. Similary, other stages of the machine learning operationalization such as the model packaging (e.g., BentoML [24], MLflow [306]), the model serving (e.g., TorchServe [55], TensorFlow Serving [270]), or the orchestration (e.g. kubeflow [63], airflow [97]) can be solved with tools that can coexist seamlessly with Orfeon.

In conclusion, Orfeon is a suitable tool for the operationalization of distributed analytical pipelines leveraging different objectives and constraints, and it advances over existing tools per the criteria in Table 4.1. In this analysis, it is clear that Orfeon excels at managing infrastructural devices present in heterogeneous environments, enforcing the different constraints inherent to the analytical domain, while being able to optimize various objectives. It promotes an automated manner to release data science projects in complex heterogeneous environments, reassuring at the same time that the business constraints and objectives are guaranteed. In fact, Orfeon has far exceeded the quality of the solutions provided by four experts in the field of ML operationalization as showcased in Table 4.5.

4.9 Conclusions and Future Work

In this research, we propose a system called Orfeon which is able to aid the operations team in the operationalization of data science projects. It comes alongside the metric shipper, a daemon capable to retrieve the necessary information and metadata from heterogeneous and distributed infrastructural devices. Orfeon consumes an analytical pipeline defined by the data scientists, and balances conflicting objectives and constraints to yield an optimal deployment specification. In addition, the user is not compelled to understand the underlying complexities of the operationalization of distributed applications on real word ecosystems. Some of these challenges include being able to utilize the benefits of multi layer ecosystems (e.g. cloud, edge) while minimizing their complexities, understanding the necessary technologies and subtleties required for such complex ecosystems, and utilizing the myriad of heterogeneous infrastructural devices that can be found in modern data ecosystems. Summarizing, the use of Orfeon enable data scientists the possibility of focusing on the development and definition of the analytical pipelines, hence making it possible to reach the overarching goal of raising the success rate of data science projects in production environments. As part of this effort, we have proposed an scenario to measure the suitability of Orfeon for the operationalization of machine learning projects leveraging different objectives. In this scenario, we have found that Orfeon is able to scale appropriately both in terms of time and memory under different

workloads, and is able to orchestrate deployments more efficiently than those produced by experts in the field. Due to this, we have concluded that, unlike the rest of alternatives under study, it is able to fulfill the criteria defined in Table 4.1, making it appropriate for the case under study.

As for the future work, this research opens up four main branches. Firstly, at the moment Orfeon is unable to handle the complexities of 5G networks which has become a hot topic recently. In this scenarios, Orfeon would need to adapt to new complex use cases such as Industrial MIoT or critical systems, promoted by the advancements in 5G technologies and the rapid adoption of MEC (Multi access Edge Computing) architectures to implement Edge Computing environments. Secondly, the monitoring and actuation of the deployed solutions is key for the success of data science projects in production environments. This is something Orfeon is lacking, and it has become our goal to incorporate such functionalities to the system to provide a better adoption by industry. Thirdly, given the overwhelming computational power of the infrastructural devices that Orfeon is able to orchestrate, it is our desire to better utilize them to obtain better results by using a distributed version of the algorithm in hand. Finally, even though Orfeon yields a goal-driven deployment specification, it does not actually perform the deployment. We aspire to utilize this definition to orchestrate the actual deployment of the given analytical pipeline on infrastructural devices spanning accross the edge and cloud layers.

I was born not knowing and have had only a little time to change that here and there.

Richard P. Feynman

5

Conclusions and Future Work

THE overarching goal of this dissertation is to promote the adoption of data science endeavors in both industry and academia by providing tools and methodologies that reinforce the ML life cycle depicted in Figure 1.1. This chapter provides a summary of the outcomes of this effort and aspires to provide the reader with a better understanding of the keys to address the problem at hand, alongside tools and methodologies to overcome the myriad of challenges in AI operationalization.

The rest of this chapter is organized in the following manner: Section 5.1 provides a detailed summary of the work and the conclusions resulting from this effort. The main contributions of this dissertation are presented in Section 5.2. Next, we revisit the hypothesis and objectives described in Chapter 1 and analyze its fulfillment in Section 5.3. The scientific publications that have led to the development of this dissertation and have served as guidance and validation for this work are listed in Section 5.4. We aspire that this research will pave the way for further advancements in the field, and we present this future work in Section 5.5. We finalize this chapter with some final remarks in Section 5.6.

5.1 Summary of Work and Conclusions

This dissertation highlights the importance of better understanding the challenges associated with the operationalization of data science projects in production environments. In this regard, the design and implementation of tools and methodologies to oversee the various stages of the ML life cycle promote the adoption of ML-based solutions by both industry and academia.

Chapter 2 sheds light on the innovative MLOps and AIOps methodologies by conducting a SLR. The former aspires to streamline the process of operationalizing ML-based projects reliably and efficiently in production environments, whereas the latter utilizes AI techniques to identify and solve common problems in IT operations. For the initial set of articles included in this SLR, distinct search terms have been identified by experts in the field. Then, various screening processes were conducted to obtain the list of studies that comprise this review. Finally, these articles have been used to shed light on the research questions defined, which have been specifically tailored to understand the challenges, opportunities, and future trends of MLOps and AIOps methodologies, and to identify the frameworks and fields in which these methodologies are trending.

Chapter 3 recognizes the necessity to clearly specify the various elements that encompass the ML life cycle in distributed analytical pipelines. For this reason, distinct DSL technologies in the field of machine learning operationalization and definition are discussed in order to establish their benefits and drawbacks. A complete DSL specification schema is offered, abstracting data scientists from the underlying technologies and technicalities that are required for the operationalization of the data science project. In this regard, PADL provides means for the data management, deployment, monitoring, and self-healing of the operationalized analytical pipeline. In addition, its descriptive nature promotes collaboration between the various teams that coexist in the project and spreads knowledge between them. On top of this, the architecture for packaging, orchestration, deployment, and monitoring leveraging PADL is designed, and various tools for the development and operationalization of ML endeavors are offered. Finally, the appropriateness of the language is discussed, considering two different use cases from independent authors.

The operationalization of distributed analytical pipelines is discussed in Chapter 4. It builds on the foundations of the previous chapter, as it utilizes the PADL specification for the description of data science projects and provides a framework coined Orfeon for the goal-driven orchestration of distributed analytical pipelines. The optimization is targeted towards the resiliency, performance, cost, and networking of the deployment; and yields an appropriate solution in a timely manner. In order to do so, an extensible edge agent devoted to gathering the necessary metrics for the optimization is also provided. On top of the regular

metrics such as CPU and memory, the agent can gather benchmark-driven metrics such as the consumption, resilience, performance, and latency of the utilized infrastructural devices. A scalability and objective analysis are conducted on the ecosystem, the former validates the suitability of the solution in terms of time and memory for various infrastructural devices and pipelines, whereas the latter validates the convergence of the distinct objectives. Finally, an experiment is conducted with various experts in the field, in which Orfeon is able to overcome the solutions provided by the experts.

In conclusion, the in-depth analysis of the state of the art in MLOps and AIOps methodologies provided in Chapter 2 led us to identify the stages that best contribute to reinforcing various stages of the ML life cycle. It can be claimed that the conjunction of the architecture and the PADL DSL showcased in Chapter 3, with the orchestration framework and agent described in Chapter 4, accomplishes the objectives of this dissertation. In the next section, the contributions of this effort and their validation are described in further detail.

5.2 Contributions

The various contributions described in Section 1.4 have been addressed throughout this dissertation. Each contribution is nested under a specific goal, as depicted in Figure 1.3. We revisit them and elaborate on how they have been tackled in this dissertation. First, the scientific contributions are presented in the following lines:

SC1 *An in-depth description stemming from the state of the art regarding the challenges and opportunities to consider for the adoption of MLOps and AIOps practices and methodologies, that aspires to boost the adoption of such methodologies in both industry and research.* This scientific contribution is nested under O1 in Chapter 2 and has been published in a Q1 scientific journal [81]. After analyzing the state of the art, we describe the open issues and challenges surrounding the adoption of MLOps and AIOps methodologies and the opportunities and future trends these practices offer to industry and academia.

SC2 *An in-depth description stemming from the state of the art regarding the frameworks, fields, and future trends in which MLOps and AIOps are thriving, that aspires to serve as reference for future implementations regarding these methodologies.* This scientific contribution results from the work pursued in O1 and is presented in Chapter 2, where the frameworks and architectures that facilitate various stages of MLOps and AIOps methodologies are presented. In addition, an analysis of current and future fields in which these practices are thriving permits determining the areas in which they are applied and considered. It has also been published in a Q1 scientific journal [81].

SC3 *A schema for the conceptualization of data science pipelines that span across the edge and cloud computational layers. It provides means for the description, monitoring, and deployment of AI projects; and covers various stages of the ML life cycle.* This contribution is nested under O2, is presented in Chapter 3, and has been published in scientific literature [89, 79]. The schema displayed in Figure 3.6 facilitates the description of heterogeneous analytical pipelines by data scientists without diving into deployment specific details, comprises the required information for the goal-driven optimization leveraging an existing infrastructure, and promotes a streamlined deployment in production environments.

SC4 *The mathematical formulation required for the operationalization optimization of data science projects. The following goals that have been considered in this research; cost, network and model performance, and resilience have been modeled.* This scientific contribution results from the work undergone for the achievement of O4. The mathematical formulation for these four fitness functions is provided in Chapter 4 and has been published in a Q1 scientific journal [80]. In addition, this formulation includes various constraints for solving the goal-driven deployment of distributed analytical pipelines. This definition promotes further implementations of this problem than the one provided as part of this dissertation.

SC5 *The future research directions stemming from the conclusions and the work presented in this dissertation.* This contribution is part of O5. One of the primary objectives of this thesis is to inspire other researchers to push the boundaries explained in this document. There have already been published studies providing state-of-the-art tools and methodologies on top of the contributions of this dissertation [194, 193], and we aspire that the conclusions depicted in Chapter 5 will continue to do so.

Next, this dissertation has also yielded various technical contributions that are revisited in detail along the following lines:

TC1 *An application for the automated retrieval of scientific literature based on predefined queries. It has been specifically tailored for an automated retrieval of the record identification in a SLR.* A framework for the automated retrieval of manuscripts that comply with various predefined search queries is provided in Chapter 2. It promotes a streamlined approach for the development of SLR and has been utilized to fulfill O1. All the details about its utilization in further manuscripts can be found on GitHub [75].

TC2 *A DSL coined PADL for the conceptualization and operationalization of distributed analytical pipelines. It has been implemented utilizing JSON Schema [84], which*

facilitates its integration with various stages of the ML life cycle. This technical contribution corresponds to O2, is explained in detail in Chapter 3, and is publicly available on GitHub [77]. The overarching goal of this technical contribution is to provide a DSL specifically tailored for data science projects to describe and model heterogeneous distributed analytical pipelines. This way, it is possible to operationalize such projects quicker, with higher quality, and narrow the gap between the various teams that participate in this endeavor.

TC3 *A PADL online validator and formatter which permits early adopters to directly type their pipelines and integrate them into their operationalization processes. Details about this implementation can be found in Chapter 3.* This contribution is nested under O2 and explained thoroughly in Chapter 3. PADL constitutes an innovative manner for data scientists to define their pipelines, hence tools and methodologies to support its adoption become necessary across the various stages that comprise the ML life cycle. The full implementation and instructions for its use are available on GitHub [77].

TC4 *A lightweight agent suitable for edge infrastructural devices coined metric shipper, which is not only able to retrieve fundamental metrics (e.g., CPU, memory), but also execute specific benchmarks specifically tailored for the data science domain (e.g., AI-Benchmark [140]). In addition, it has been developed to be cross-platform to be able to adapt to the heterogeneous infrastructural devices found in the edge.* This technical contribution refers to O3, the agent has been implemented and tested in the heterogeneous infrastructural devices presented in Chapter 4 and can collect the various metrics displayed in Table 4.2, which are fed into subsequent stages of the ML life cycle. The source code is available for public use on GitHub [73].

TC5 *A pipeline orchestration framework coined Orfeon, which has been designed on top of the functionality offered by the PADL DSL. It provides means for the goal-driven optimization of data science pipelines across the various devices that can be found in the cloud and edge computational layers.* All the details about the development can be found in Chapter 4. This contribution is the cornerstone of O4 and is publicly available on GitHub [73]. It builds on the foundations established by all the previous in order to provide a goal-driven operationalization of heterogeneous distributed analytical pipelines. The design, implementation, and validation of this effort can be found in Chapter 4. This effort paved the way for applying for a patent in the field of deployment optimization [273].

5.3 Hypothesis and Objective Validation

The following hypothesis, which was first introduced in Section 1.2, has served as the cornerstone for the development of this dissertation:

> **Hypothesis.** *Using innovative Artificial Intelligence-driven tools and technologies to oversee the various stages of the machine learning lifecycle outperforms the results provided by experts in the field of Machine Learning operations.*

In order to validate this hypothesis, the following goal was defined and is shown below for convenience:

> **Goal.** *To design and implement a domain-specific language (DSL) for the description of distributed analytical pipelines, which can be fed into a goal-driven operationalization framework for heterogeneous environments operated by complex metrics.*

However, accurately quantifying the degree of fulfillment of such a broad challenge is cumbersome, and more specific and measurable objectives were identified in Section 1.2. In the next few lines, the way these individual objectives have been addressed throughout this dissertation is going to be scrutinized to evaluate their fulfillment.

O1 *To provide insights into the challenges and benefits for the adoption of MLOps and AIOps methodologies in order to promote the adoption of AI-based solutions in both industry and academia.* This goal is thoroughly addressed in Chapter 2. We have conducted an SLR that provides insights for the adoption of MLOps and AIOps methodologies in both industry and academia. In particular, the SLR elaborates on five research questions and performs a qualitative analysis of the selected manuscripts.

O2 *The analysis, design and implementation of a description methodology for distributed analytical pipelines that oversees the definition, deployment and monitoring of heterogeneous MLOps use cases.* Chapter 3 analyzes the state of the art in domain-specific languages focused on the analytical domain. This investigation leads to the design of a schema and the implementation of a DSL specifically tailored to be utilized within the analytical domain, with a special emphasis on distributed architectures. The purpose of it is to alleviate the burden of deploying data science projects in production environments.

O3 *The design and implementation of a framework that is able to leverage conflicting objectives for the operationalization of distributed analytical pipelines.* In Chapter 4,

the rationale for the implementation of such a framework is covered. In addition, details regarding the design and implementation of the various components that comprise the orchestrator are explored. Then, an experimental setup validates the appropriateness of the system under varying conditions. Finally, an expert evaluation is conducted in order to assess the benefits of the framework.

O4 *The design and implementation of an edge service that is able to evaluate heterogeneous infrastructural devices to obtain the necessary information for a near optimal deployment.* This objective and its rationale have been addressed in Chapter 4. The unique necessities imposed by the framework implemented because of the previous goal, in conjunction with the heterogeneous infrastructural devices that comprise the edge computational layer, have led to the design and development of an edge service that is able to operate on the heterogeneous infrastructural devices found in the edge computational layer. It can gather complex metrics that are essential for the goal-driven operationalization of distributed analytical pipelines.

05 *To state the conclusions of this dissertation in order to identify future research inspired by the work undergone in this thesis.* The conclusions of this dissertation are presented in Chapter 5. Furthermore, research directions that have been identified throughout this thesis are also presented, since we aspire they will serve as cornerstones for the future.

For further clarity, the relationship between the chapters of this dissertation, the aforementioned objectives, and the technical and scientific contributions presented in the previous subsection is shown in Figure 5.1.

The completion of these individual objectives results in the fulfillment of the main goal of this dissertation, presented in Section 1.2. Accordingly, the hypothesis of this research presented in the same section becomes confirmed. A streamlined pipeline comprising the PADL DSL designed and implemented in Chapter 3 and the Orfeon orchestrator developed within Chapter 4 has been implemented and validated. Thus, we expect that the work developed within this dissertation will inspire the adoption of data science projects in both industry and academia, since it widens the state of the art on challenges and pitfalls in MLOps and AIOps practices and methodologies and provides tools to oversee various stages of the ML life cycle.

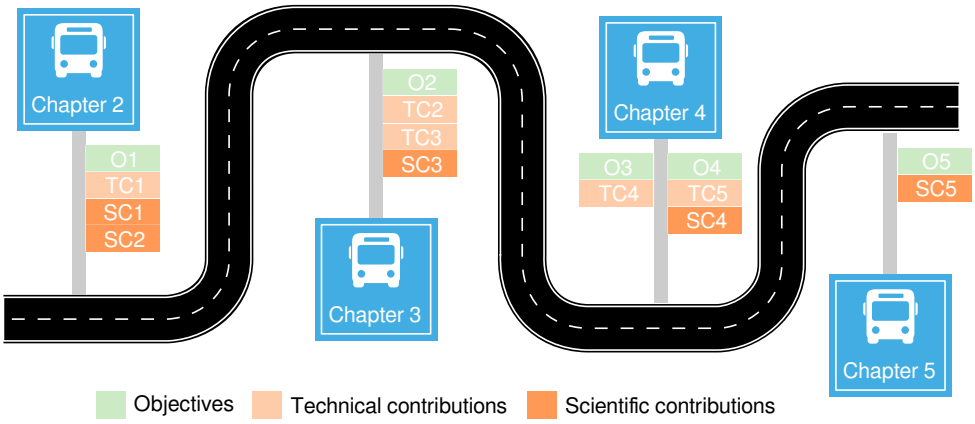

Fig. 5.1 Dissertation roadmap showcasing the relationship between the chapters of this dissertation, the objectives, and the scientific and technical contributions.

5.4 Relevant Publications

Over the years, academic experts have played a crucial role in directing the course of this research. This feedback and guidance have been obtained through scientific publications. In what follows, the list of published manuscripts in JCR journals, international conferences, and collaborations with other projects and groups is shown.

5.4.1 International JCR journals

The complete implementation of the PADL DSL, which served as the foundation for the remainder of the work presented in this dissertation, was published in the following journal:

- Díaz-de-Arcaya, J., Miñón, R., Torre-Bastida, A. I., Del Ser, J., & Almeida, A. (2020). PADL: A modeling and deployment language for advanced analytical services. Sensors, 20(23), 6712.

The conceptualization and implementation of the goal-driven orchestrator coined Orfeon were first presented in the following journal:

- Díaz-de-Arcaya, J., Torre-Bastida, A. I., Miñón, R., & Almeida, A. (2023). Orfeon: An AIOps framework for the goal-driven operationalization of distributed analytical pipelines. Future Generation Computer Systems, 140, 18-35.

A systematic literature review conceived to shed light on the adoption of MLOps and AIOps methodologies in both industry and academia was published in the following journal:

- Diaz-de-Arcaya, J., Torre-Bastida, A. I., Zárate, G., Miñón, R., & Almeida, A. (2023). A Joint Study of the Challenges, Opportunities, and Roadmap of MLOps and AIOps: A Systematic Survey. ACM Computing Surveys.

5.4.2 International conferences

A preliminary implementation of the PADL DSL which led to the research done in this dissertation, was presented at the following conference:

- Díaz-de-Arcaya, J., Miñón, R., Torre-Bastida, A. I., Del Ser, J., & Almeida, A. (2020, September). Padl: a language for the operationalization of distributed analytical pipelines over edge/fog computing environments. In 2020 5th International Conference on Smart and Sustainable Technologies (SpliTech) (pp. 1-6). IEEE.

A framework devoted to the operationalization that encompasses the various stages and technologies involved in the application life cycle is presented at the following conference:

- Diaz-de-Arcaya, J., Osaba, E., Benguria, G., Etxaniz, I., L. Lobo, J., Alonso, J., ... & Almeida, A. (2023, April). IEM: A Unified Lifecycle Orchestrator for Multilingual IaC Deployments. In Companion of the 2023 ACM/SPEC International Conference on Performance Engineering (pp. 195-199).

A meta-system for the resilient optimization of workloads in multilayer architectures leveraging the edge computing layer is presented at the following conference:

- Diaz-de-Arcaya, J., Torre-Bastida, A. I., Bonilla, L., López-de-Armentia, J., Miñón, R., Zarate, G., & Almeida, A. (2023, June). Akats: A System for Resilient Deployments on Edge Computing Environments Using Federated Machine Learning Techniques. In 2023 8th International Conference on Smart and Sustainable Technologies (SpliTech) (pp. 1-4). IEEE.

5.4.3 Patents

A patent application in the realm of analytical pipeline optimization was filed with the European Patent Office:

- Torre-Bastida, A. I., Diaz-de-Arcaya, J., Miñon, R, & Almeida, A. (2022, September). Method, system and computer program product for optimizing the deployment of analytical pipelines. EP 4,064,047 A1. European Patent Office.

5.4.4 Collaborations

The following publications are the results of the collaborations that have taken place throughout the development of this dissertation and have been included in this subsection as they represent innovative research that is directly related to the topic of this thesis.

- Alonso, J., Orue-Echevarria, L., Osaba, E., López Lobo, J., Martinez, I., Diaz de Arcaya, J., & Etxaniz, I. (2021). Optimization and Prediction Techniques for Self-Healing and Self-Learning Applications in a Trustworthy Cloud Continuum. Information, 12(8), 308.

- Torre-Bastida, A. I., Díaz-de-Arcaya, J., Osaba, E., Muhammad, K., Camacho, D., & Del Ser, J. (2021). Bio-inspired computation for big data fusion, storage, processing, learning and visualization: state of the art and future directions. Neural Computing and Applications, 1-31.

- Torre-Bastida, A. I., Gil, G., Miñón, R., & Díaz-de-Arcaya, J. (2022). Technological Perspective of Data Governance in Data Space Ecosystems. In Data Spaces (pp. 65-87). Springer, Cham.

- Zárate, G., Miñón, R., Díaz-de-Arcaya, J., & Torre-Bastida, A. I. (2022, March). K2E: Building MLOps Environments for Governing Data and Models Catalogues while Tracking Versions. In 2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C) (pp. 206-209). IEEE.

- Miñón, R., Diaz-de-Arcaya, J., Torre-Bastida, A. I., & Hartlieb, P. (2022). Pangea: An MLOps Tool for Automatically Generating Infrastructure and Deploying Analytic Pipelines in Edge, Fog and Cloud Layers. Sensors, 22(12), 4425.

- Miñón, R., Díaz-de-Arcaya, J., Torre-Bastida, A. I., Zarate, G., & Moreno-Fernandez-de-Leceta, A. (2022, July). MLPacker: A Unified Software Tool for Packaging and Deploying Atomic and Distributed Analytic Pipelines. In 2022 7th International Conference on Smart and Sustainable Technologies (SpliTech) (pp. 1-6). IEEE.

- Osaba, E., Diaz-de-Arcaya, J., Orue-Echevarria, L., Alonso, J., Lobo, J. L., Benguria, G., & Etxaniz, I. (2022, July). PIACERE project: description and prototype for optimizing infrastructure as code deployment configurations. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (pp. 71-72).

- Osaba, E., Diaz-de-Arcaya, J., Alonso, J., Lobo, J. L., Benguria, G., & Etxaniz, I. (2023, February). An Evolutionary Computation-Based Platform for Optimizing

Infrastructure-as-Code Deployment Configurations. In International Congress on Information and Communication Technology (pp. 321-330). Singapore: Springer Nature Singapore.

5.5 Future Work

Inspired by the work presented in this dissertation and its limitations, the following research lines and future work have been identified:

- In this dissertation, we have focused on the modeling and orchestration of distributed analytical pipelines. Accordingly, the deployment of the solution itself has been deliberately left out of the scope of this dissertation. In this regard, we believe that this stage requires research attention as it is a complex process encompassing various stages. In particular, the deployment needs to consider the provisioning of the infrastructural devices required, the networking, the configuration of these devices to comply with the requirements, and the deployment of the business elements required for the project. To the best of our knowledge, no technology or framework can comply with these stages in a unified manner, and the myriad technologies available make this process cumbersome at best. In addition, the operationalization of the business logic needs to leverage not only the deployment of the application but also the data management, networking challenges, packaging, and security concerns. Thus, we expect significant advances to take place in this field in the foreseeable future.

- In Chapter 4, we have presented a framework for the goal-driven operationalization of distributed analytical pipelines. We have focused on the cost, model performance, resiliency, and networking of data science solutions. However, we believe there is plenty of room for further developing the networking optimization of the solutions. The recent advances brought by technologies such as 5G in the form of software-defined networking (SDN) enable the integration of advanced networking optimizations in Orfeon. In addition, innovative technologies such as Open Source MANO [1], which provides a production-quality framework stack to orchestrate and consume Network Functions Virtualization (NFS) architectures, provide means for the provisioning of the networking required by distributed analytical pipelines based on the optimization yielded by Orfeon.

- The edge based agent presented in Chapter 4, coined metric shipper, has been specifically tailored to provide the operationalization framework with the necessary metrics

[1] https://osm.etsi.org/

for the optimization. In this regard, the metric shipper provides system information such as CPU and memory, and benchmark-driven metrics such as consumption, performance, and latency. It would be beneficial to expand the functionality of the agents to cover various dimensions of the architecture (e.g., infrastructure, model, data validation, software security). In addition, an online prediction of the gathered metrics would open up the possibility of implementing an alert-driven self-healing scenario, in which the agents could foresee possible issues and trigger healing scenarios if the problem could be solved locally, or an alert to the architecture otherwise.

- The combination of the DSL presented in Chapter 3, and the goal-driven framework and agent presented in Chapter 4 pave the way for the implementation of a self-driven and self-healing architecture in which the issues that are raised are automatically addressed without the manual intervention of the end user. However, pursuing this endeavor requires deep research and implementation efforts across all the components. For instance, innovative monitoring and triggering strategies need to be considered in PADL and Orfeon, the agents would be required to gather advanced metrics, and new elements would be required to be incorporated into the architecture to support real-time messaging and actuation capabilities.

- The framework described in Chapter 4, coined Orfeon, yields a set of solutions that comply with the constraints imposed by the problem and leverage the various objectives modeled by the fitness functions. While ranking the solutions in a single-objective problem does not represent a challenge, choosing the best solution in a multidimensional problem requires a more complex evaluation. Objective prioritization and fitness normalization are among the myriad techniques that can be utilized to select the single best solution. However, an in-depth analysis would benefit the state of the art and complement the functionality of Orfeon. On the other hand, the termination criteria is yet another research line that may attract attention, as more elaborate and dynamic formulations could be utilized for finishing the formulation, such as utilizing the tangent slope or when a predefined percentage of the normalized objective is attained.

- Certain stages of the ML life cycle are not yet covered in the solution presented in this dissertation. For instance, training and inference have been deliberately left out of the final product. In this regard, the DSL covered in Chapter 3 could benefit from other technologies for the description of data science applications, such as PFA [227], PMML [111], or TOSCA [26].

- The recent upsurge of LLMs has the potential to enhance MLOps and AIOps methodologies by streamlining the process of test generation and coverage within the Machine Learning lifecycle. This can increase error detection in preliminary stages, reduce human effort, and improve accuracy in developing and deploying ML models.
- The success of artificial intelligence heavily depends on the quality of the data it utilizes. The World Wibe Web Consortium classifies the various dimensions of data quality in three categories: inherent, system-dependent, and a combination of both [290]. In this regard, artificial intelligence can assist in the process of categorizing, cleaning, and enhancing data quality.

5.6 Final Remarks

This work was envisioned as a way to promote and facilitate the adoption of AI by both academia and industry. In order to do so, we leverage the ML life cycle to provide methodologies and tools that assist data scientists and IT operations in the operationalization of data science projects in production environments. We hope that the contributions of this dissertation and the future work stated will guide other researchers into pushing the boundaries of this research area.

References

[1] U. Aguilera, O. Peña, O. Belmonte, and D. López-de Ipiña. Citizen-centric data services for smarter cities. *Future Generation Computer Systems*, 76:234–247, 2017.

[2] S. Ahmed, M. Singh, B. Doherty, E. Ramlan, K. Harkin, and D. Coyle. Ai for information tecchnology operation (aiops): A review of it incident risk prediction. In *2022 9th International Conference on Soft Computing & Machine Intelligence (ISCMI)*, pages 253–257. IEEE, 2022.

[3] Z. Ahmed, S. Amizadeh, M. Bilenko, R. Carr, W.-S. Chin, Y. Dekel, X. Dupre, V. Eksarevskiy, S. Filipi, T. Finley, et al. Machine learning at microsoft with ml. net. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2448–2458, 2019.

[4] E. Al Nuaimi, H. Al Neyadi, N. Mohamed, and J. Al-Jaroodi. Applications of big data to smart cities. *Journal of Internet Services and Applications*, 6(1):25, 2015.

[5] M. Alshangiti, H. Sapkota, P. K. Murukannaiah, X. Liu, and Q. Yu. Why is developing machine learning applications challenging? a study on stack overflow posts. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–11. IEEE, 2019.

[6] F. Alves, H. Badikyan, H. A. Moreira, J. Azevedo, P. M. Moreira, L. Romero, and P. Leitão. Deployment of a smart and predictive maintenance system in an industrial case study. In *2020 IEEE 29th International Symposium on Industrial Electronics (ISIE)*, pages 493–498. IEEE, 2020.

[7] J. M. Alves, L. M. Honório, and M. A. Capretz. Ml4iot: A framework to orchestrate machine learning workflows on internet of things data. *IEEE Access*, 7:152953–152967, 2019.

[8] I. Amazon Web Services. Amazon sagemaker. https://aws.amazon.com/sagemaker/, 2022. Last accessed 14 March 2022.

[9] I. Amazon Web Services. Cloud computing services - amazon web services (aws). https://aws.amazon.com/, 2022. Last accessed 14 March 2022.

[10] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann. Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 291–300. IEEE, 2019.

[11] H. Arasteh, V. Hosseinnezhad, V. Loia, A. Tommasetti, O. Troisi, M. Shafie-Khah, and P. Siano. Iot-based smart cities: A survey. *EEEIC 2016 - International Conference on Environment and Electrical Engineering*, pages 2–7, 2016.

[12] E. Aromataris and A. Pearson. The systematic review: an overview. *AJN The American Journal of Nursing*, 114(3):53–58, 2014.

[13] M. Arostegi, A. Torre-Bastida, M. N. Bilbao, and J. Del Ser. A heuristic approach to the multicriteria design of iaas cloud infrastructures for big data applications. *Expert Systems*, 35(5):e12259, 2018.

[14] R. Ashmore, R. Calinescu, and C. Paterson. Assuring the machine learning lifecycle: Desiderata, methods, and challenges. *ACM Computing Surveys (CSUR)*, 54(5):1–39, 2021.

[15] F. Assunção, N. Lourenço, B. Ribeiro, and P. Machado. Evolution of scikit-learn pipelines with dynamic structured grammatical evolution. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pages 530–545. Springer, 2020.

[16] T. P. Bac, M. N. Tran, and Y. Kim. Serverless computing approach for deploying machine learning applications in edge layer. In *2022 International Conference on Information Networking (ICOIN)*, pages 396–401. IEEE, 2022.

[17] A. Badiola-Bengoa and A. Mendez-Zorrilla. A systematic review of the application of camera-based human pose estimation in the field of sport and physical exercise. *Sensors*, 21(18):5996, 2021.

[18] S. K. Baduge, S. Thilakarathna, J. S. Perera, M. Arashpour, P. Sharafi, B. Teodosio, A. Shringi, and P. Mendis. Artificial intelligence and smart vision for building and construction 4.0: Machine and deep learning methods and applications. *Automation in Construction*, 141:104440, 2022.

[19] A. Barrak, F. Petrillo, and F. Jaafar. Serverless on machine learning: A systematic mapping study. *IEEE Access*, 2022.

[20] M. Barry, S. El Jaouhari, A. Bifet, J. Montiel, E. Guerizec, and R. Chiky. Streamflow: A system for summarizing and learning over industrial big data streams. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 2198–2205. IEEE, 2022.

[21] H. Bastani, D. J. Zhang, and H. Zhang. Applied machine learning in operations management. In *Innovative Technology at the Interface of Finance and Operations*, pages 189–222. Springer, 2022.

[22] O. Benfeldt, J. S. Persson, and S. Madsen. Data governance as a collective action problem. *Information Systems Frontiers*, 22:299–313, 2020.

[23] A. Benitez-Hidalgo, A. J. Nebro, J. Garcia-Nieto, I. Oregi, and J. Del Ser. jmetalpy: A python framework for multi-objective optimization with metaheuristics. *Swarm and Evolutionary Computation*, 51:100598, 2019.

[24] BentoML. Bentoml. https://www.bentoml.com/, 2022. Last accessed 14 March 2022.

[25] A. Bhattacharjee, Y. Barve, S. Khare, S. Bao, A. Gokhale, and T. Damiano. Stratum: A serverless framework for the lifecycle management of machine learning-based data analytics tasks. In *2019 {USENIX} Conference on Operational Machine Learning (OpML 19)*, pages 59–61, 2019.

[26] T. Binz, U. Breitenbücher, O. Kopp, and F. Leymann. Tosca: portable automated deployment and management of cloud applications. In *Advanced Web Services*, pages 527–549. Springer, 2013.

[27] J. Blank and K. Deb. pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

[28] C. Bormann, A. P. Castellani, and Z. Shelby. Coap: An application protocol for billions of tiny internet nodes. *IEEE Internet Computing*, pages 62–67, 2012.

[29] O. Boursalie, R. Samavi, and T. E. Doyle. Machine learning and mobile health monitoring platforms: a case study on research and implementation challenges. *Journal of Healthcare Informatics Research*, 2(1):179–203, 2018.

[30] H. B. Braiek and F. Khomh. On testing machine learning programs. *Journal of Systems and Software*, 164:110542, 2020.

[31] D. Brayford and S. Vallecorsa. Deploying scientific al networks at petaflop scale on secure large scale hpc production systems with containers. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, pages 1–8, 2020.

[32] D. Brayford, S. Vallecorsa, A. Atanasov, F. Baruffa, and W. Riviera. Deploying ai frameworks on secure hpc systems with containers. In *2019 ieee high performance extreme computing conference (hpec)*, pages 1–6. IEEE, 2019.

[33] E. Brynjolfsson, D. Rock, and C. Syverson. Artificial intelligence and the modern productivity paradox: A clash of expectations and statistics. In *The economics of artificial intelligence: An agenda*, pages 23–57. University of Chicago Press, 2018.

[34] A. Burrello, A. Garofalo, N. Bruschi, G. Tagliavini, D. Rossi, and F. Conti. Dory: Automatic end-to-end deployment of real-world dnns on low-cost iot mcus. *IEEE Transactions on Computers*, 70(8):1253–1268, 2021.

[35] E. B.V. Beats: Data shippers for elasticsearch. https://www.elastic.co/beats/, 2022. Last accessed 8 March 2022.

[36] E. B.V. Elsevier developer portal, 2022. Last accessed 19 June 2022.

[37] L. Bălan, K. Kunii, L. Hoang, A. Ivaniuk, D. Deriabin, Y. Dada, D. Datta, Z. Patel, I. Danov, G. Wrigley, J. Stichbury, N. Khan, M. Theisen, N. Tsaousis, W. Walker, T. Nguyen, R. Westenra, L. Carvalho, M. D. Trevisani, S. Bertoli, S. Mawjee, sasaki takeru, A. Milne, B. Nijholt, D. Vukolov, K. Fischer, M. Koops, Vijaykumar, and Y. Minami. quantumblacklabs/kedro: 0.17.1. https://doi.org/10.5281/zenodo.4581118, Mar. 2021.

[38] F. Calefato, F. Lanubile, and L. Quaranta. A preliminary investigation of mlops practices in github. In *Proceedings of the 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 283–288, 2022.

[39] M. W. Camargo, M. S. Serpa, D. Carastan-Santos, A. Carissimi, and P. O. Navaux. Accelerating machine learning algorithms with tensorflow using thread mapping policies. In *Latin American High Performance Computing Conference*, pages 62–70. Springer, 2020.

[40] A. Campion, M.-G. Hernandez, S. M. Jankin, and M. Esteve. Managing artificial intelligence deployment in the public sector. *Computer*, 53(10):28–37, 2020.

[41] M. Caramia and P. Dell'Olmo. *Multi-objective management in freight logistics*. Springer, 2008.

[42] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas. Apache flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 36(4), 2015.

[43] J. Carreira, P. Fonseca, A. Tumanov, A. Zhang, and R. Katz. Cirrus: A serverless framework for end-to-end ml workflows. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 13–24, 2019.

[44] J. Carter. speedtest-cli. https://packages.ubuntu.com/source/bionic/speedtest-cli, 2022. Last accessed 25 July 2022.

[45] M. Cerchecci, F. Luti, A. Mecocci, S. Parrino, G. Peruzzi, and A. Pozzebon. A low power iot sensor node architecture for waste management within smart cities context. *Sensors*, 18(4):1282, 2018.

[46] D. Chahal, R. Ojha, M. Ramesh, and R. Singhal. Migrating large deep learning models to serverless architecture. In *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 111–116. IEEE, 2020.

[47] R. Chen, Y. Pu, B. Shi, and W. Wu. An automatic model management system and its implementation for aiops on microservice platforms. *The Journal of Supercomputing*, 79(10):11410–11426, 2023.

[48] Q. Cheng, D. Sahoo, A. Saha, W. Yang, C. Liu, G. Woo, M. Singh, S. Saverese, and S. C. Hoi. Ai for it operations (aiops) on cloud platforms: Reviews, opportunities and challenges. *arXiv preprint arXiv:2304.04661*, 2023.

[49] G. Chiandussi, M. Codegone, S. Ferrero, and F. E. Varesio. Comparison of multi-objective optimization methodologies for engineering applications. *Computers & Mathematics with Applications*, 63(5):912–942, 2012.

[50] V. R. Chintapalli, K. Kondepu, A. Sgambelluri, B. R. Tamma, P. Castoldi, L. Valcarenghi, et al. Orchestrating edge-and cloud-based predictive analytics services. In *2020 European Conference on Networks and Communications (EuCNC)*, pages 214–218. IEEE, 2020.

[51] M. Ciavotta, D. Ardagna, and G. P. Gibilisco. A mixed integer linear programming optimization approach for multi-cloud capacity allocation. *Journal of Systems and Software*, 123:64–78, 2017.

[52] Z. M. Çınar, A. Abdussalam Nuhu, Q. Zeeshan, O. Korhan, M. Asmael, and B. Safaei. Machine learning in predictive maintenance towards sustainable smart manufacturing in industry 4.0. *Sustainability*, 12(19):8211, 2020.

[53] CISOfy. Lynis. https://cisofy.com/lynis/, 2021. Last accessed 7 March 2022.

[54] F. comissioned by Google. Modernize with aiops to maximize your impact, 2023. Last accessed 29 August 2023.

[55] P. S. Contributors. Torchserve. https://pytorch.org/serve/, 2022. Last accessed 14 March 2022.

[56] D. Cournapeau. Scikit-learn. https://scikit-learn.org, 2020. Last accessed 21 September 2023.

[57] Crossref. Crossref api documentation, 2022. Last accessed 19 June 2022.

[58] A. Dakkak, C. Li, J. Xiong, and W.-m. Hwu. Dlspec: A deep learning task exchange specification. In *2020 {USENIX} Conference on Operational Machine Learning (OpML 20)*, 2020.

[59] G. Dandachi, A. De Domenico, D. T. Hoang, and D. Niyato. An artificial intelligence framework for slice deployment and orchestration in 5g networks. *IEEE Transactions on Cognitive Communications and Networking*, 6(2):858–871, 2019.

[60] Y. Dang, Q. Lin, and P. Huang. Aiops: real-world challenges and research innovations. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 4–5. IEEE, 2019.

[61] T. H. Davenport. From analytics to artificial intelligence. *Journal of Business Analytics*, 1(2):73–80, 2018.

[62] T. H. Davenport and R. Bean. Big data and ai executive survey (2019). *NewVantage Partners (NVP), Tech. Rep*, 2019.

[63] V. K. David Aronchick, Jeremy Lewi. Kubeflow. https://www.kubeflow.org/, 2022. Last accessed 13 March 2022.

[64] J. D. de Arcaya. Padl-cli. https://hub.docker.com/r/josuarcaya/padl-cli, 2020. Last accessed 20 September 2023.

[65] J. D. de Arcaya. Padl-web. https://hub.docker.com/r/josuarcaya/padl-web, 2020. Last accessed 20 September 2023.

[66] E. De Coninck, S. Bohez, S. Leroux, T. Verbelen, B. Vankeirsbilck, P. Simoens, and B. Dhoedt. Dianne: a modular framework for designing, training and deploying deep neural networks on heterogeneous distributed infrastructure. *Journal of Systems and Software*, 141:52–65, 2018.

[67] M. M. de Medeiros, N. Hoppen, and A. C. G. Maçada. Data science for business: Benefits, challenges and opportunities. *The Bottom Line*, 2020.

[68] D. De Silva and D. Alahakoon. An artificial intelligence life cycle: From conception to production. *Patterns*, page 100489, 2022.

[69] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In *International conference on parallel problem solving from nature*, pages 849–858. Springer, 2000.

[70] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4):577–601, 2013.

[71] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[72] O. Debauche, S. Mahmoudi, S. A. Mahmoudi, P. Manneback, and F. Lebeau. A new edge architecture for ai-iot services deployment. *Procedia Computer Science*, 175:10–19, 2020.

[73] J. Diaz-de-Arcaya, R. Miñon, A. I. Torre-Bastida, and A. Almeida. Official Orfeon repository. https://github.com/josu-arcaya/orfeon, 2021. Last accessed 21 September 2023.

[74] J. Diaz-de Arcaya, R. Miñon, A. I. Torre-Bastida, and A. Almeida. Official Orfeon repository. https://github.com/josu-arcaya/orfeon, 2021. Last accessed 7 March 2022.

[75] J. Diaz-de-Arcaya, R. Miñon, A. I. Torre-Bastida, A. Almeida, and G. Zarate. Official SLR repository. https://github.com/josu-arcaya/slr, 2022. Last accessed 31 January 2022.

[76] J. Diaz-de Arcaya, R. Miñon, A. I. Torre-Bastida, A. Almeida, and G. Zarate. Official SLR repository, 2022. Last accessed 15 December 2022.

[77] J. Diaz-de Arcaya, R. Miñon, A. I. Torre-Bastida, J. Del Ser, and A. Almeida. Official PADL repository. https://github.com/josu-arcaya/padl, 2020. Last accessed 31 January 2022.

[78] J. Díaz-de Arcaya, R. Miñón, A. I. Torre-Bastida, J. Del Ser, and A. Almeida. Padl: a language for the operationalization of distributed analytical pipelines over edge/-fog computing environments. In *2020 5th International Conference on Smart and Sustainable Technologies (SpliTech)*, pages 1–6. IEEE, 2020.

[79] J. Díaz-de Arcaya, R. Miñón, A. I. Torre-Bastida, J. Del Ser, and A. Almeida. Padl: A modeling and deployment language for advanced analytical services. *Sensors*, 20(23):6712, 2020.

[80] J. Díaz-de Arcaya, A. I. Torre-Bastida, R. Miñón, and A. Almeida. Orfeon: An aiops framework for the goal-driven operationalization of distributed analytical pipelines. *Future Generation Computer Systems*, 140:18–35, 2023.

[81] J. Diaz-de Arcaya, A. I. Torre-Bastida, G. Zárate, R. Miñón, and A. Almeida. A joint study of the challenges, opportunities, and roadmap of mlops and aiops: A systematic survey. *ACM Computing Surveys*, 2023.

[82] Docker. Docker. https://www.docker.com/, 2022. Last accessed 14 March 2022.

[83] T. Dou, Y. K. Lopes, P. Rockett, E. A. Hathway, and E. Saber. Gpml: an xml-based standard for the interchange of genetic programming trees. *Genetic Programming and Evolvable Machines*, 21(4):605–627, 2020.

[84] M. Droettboom et al. Understanding json schema. *Available on: http://spacetelescope. github. io/understanding-jsonschema/UnderstandingJSONSchema. pdf (accessed on 14 April 2014)*, 2015.

[85] J. Du, Q. Zhu, Y. Shi, Q. Wang, Y. Lin, and D. Zhao. Cognition digital twins for personalized information systems of smart cities: Proof of concept. *Journal of Management in Engineering*, 36(2):04019052, 2020.

[86] J. J. Durillo and A. J. Nebro. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771, 2011.

[87] R. Dzhusupova, J. Bosch, and H. H. Olsson. Challenges in developing and deploying ai in the engineering, procurement and construction industry. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1070–1075. IEEE, 2022.

[88] J. Díaz-de-Arcaya, R. Miñon, and A. I. Torre-Bastida. Towards an architecture for big data analytics leveraging edge/fog paradigms. In *Proceedings of the 13th European Conference on Software Architecture-Volume 2*, pages 173–176, 2019.

[89] J. Díaz-de-Arcaya, R. Miñón, A. I. Torre-Bastida, J. Del Ser, and A. Almeida. Padl: a language for the operationalization of distributed analytical pipelines over edge/fog computing environments. In *Proceedings of the 5th International Conference on Smart and Sustainable Technologies*, 2020.

[90] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano. Devops. *Ieee Software*, 33(3):94–100, 2016.

[91] A. Engelfriet. Choosing an open source license. *IEEE software*, 27(1):48–49, 2009.

[92] G. Flamis, S. Kalapothas, and P. Kitsos. Best practices for the deployment of edge inference: The conclusions to start designing. *Electronics*, 10(16):1912, 2021.

[93] M. Folk, G. Heber, Q. Koziol, E. Pourmal, and D. Robinson. An overview of the hdf5 technology suite and its applications. In *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*, pages 36–47, 2011.

[94] A. S. Foundation. Apache brooklyn. https://brooklyn.apache.org/, 2022. Last accessed 13 March 2022.

[95] C. N. C. Foundation. Kubernetes. https://kubernetes.io/, 2022. Last accessed 13 March 2022.

[96] P. S. Foundation. pickle - python object serialization. https://docs.python.org/3/library/pickle.html, 2022. Last accessed 9 March 2022.

[97] T. A. S. Foundation. Apache airflow. https://airflow.apache.org/, 2022. Last accessed 14 March 2022.

[98] S. Frey, F. Fittkau, and W. Hasselbring. Search-based genetic optimization for deployment and reconfiguration of software in the cloud. In *2013 35th international conference on software engineering (ICSE)*, pages 512–521. IEEE, 2013.

[99] G. Furano, A. Tavoularis, and M. Rovatti. Ai in space: Applications examples and challenges. In *2020 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 1–6. IEEE, 2020.

[100] G. Fursin, H. Guillou, and N. Essayan. Codereef: an open platform for portable mlops, reusable automation actions and reproducible benchmarking. *arXiv preprint arXiv:2001.07935*, 2020.

[101] Á. L. García, J. M. De Lucas, M. Antonacci, W. Zu Castell, M. David, M. Hardt, L. L. Iglesias, G. Moltó, M. Plociennik, V. Tran, et al. A cloud-based framework for machine learning workloads and applications. *IEEE access*, 8:18681–18692, 2020.

[102] V. Garousi, M. Felderer, and M. V. Mäntylä. The need for multivocal literature reviews in software engineering: complementing systematic literature reviews with grey literature. In *Proceedings of the 20th international conference on evaluation and assessment in software engineering*, pages 1–6, 2016.

[103] G. Gharibi, V. Walunj, R. Alanazi, S. Rella, and Y. Lee. Automated management of deep learning experiments. In *Proceedings of the 3rd International Workshop on Data Management for End-to-End Machine Learning*, pages 1–4, 2019.

[104] A. Ghimire, S. Thapa, A. K. Jha, S. Adhikari, and A. Kumar. Accelerating business growth with big data and artificial intelligence. In *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, pages 441–448. IEEE, 2020.

[105] Google. Ai platform. https://cloud.google.com/ai-platform/docs, 2021. Last accessed 14 March 2022.

[106] N. K. Gopalakrishna, D. Anandayuvaraj, A. Detti, F. L. Bland, S. Rahaman, and J. C. Davis. " if security is required" engineering and security practices for machine learning-based iot devices. In *Proceedings of the 4th International Workshop on Software Engineering Research and Practice for the IoT*, pages 1–8, 2022.

[107] E. Gossett, C. Toher, C. Oses, O. Isayev, F. Legrain, F. Rose, E. Zurek, J. Carrete, N. Mingo, A. Tropsha, et al. Aflow-ml: A restful api for machine-learning predictions of materials properties. *Computational Materials Science*, 152:134–145, 2018.

[108] T. Granlund, A. Kopponen, V. Stirbu, L. Myllyaho, and T. Mikkonen. Mlops challenges in multi-organization setup: Experiences from two real-world cases. In *2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)*, pages 82–88. IEEE, 2021.

[109] S. Grigorescu, T. Cocias, B. Trasnea, A. Margheri, F. Lombardi, and L. Aniello. Cloud2edge elastic ai framework for prototyping and deployment of ai inference engines in autonomous vehicles. *Sensors*, 20(19):5450, 2020.

[110] J. Grohmann, P. K. Nicholson, J. O. Iglesias, S. Kounev, and D. Lugones. Monitorless: Predicting performance degradation in cloud applications with machine learning. In *Proceedings of the 20th international middleware conference*, pages 149–162, 2019.

[111] A. Guazzelli, M. Zeller, W.-C. Lin, G. Williams, et al. PMML: An open standard for sharing models. *The R Journal*, 1(1):60–65, 2009.

[112] V. Gudivada, A. Apon, and J. Ding. Data quality considerations for big data and machine learning: Going beyond data cleaning and transformations. *International Journal on Advances in Software*, 10(1):1–20, 2017.

[113] T. S. Gunawan, M. K. Lim, N. F. Zulkurnain, and M. Kartiwi. On the review and setup of security audit using kali linux. *Indonesian Journal of Electrical Engineering and Computer Science*, 11(1):51–59, 2018.

[114] A. Gunny, D. Rankin, P. Harris, E. Katsavounidis, E. Marx, M. Saleem, M. Coughlin, and W. Benoit. A software ecosystem for deploying deep learning in gravitational wave physics. In *Proceedings of the 12th Workshop on AI and Scientific Computing at Scale using Flexible Computing Infrastructures*, pages 9–17, 2022.

[115] Q. Guo, S. Chen, X. Xie, L. Ma, Q. Hu, H. Liu, Y. Liu, J. Zhao, and X. Li. An empirical study towards characterizing deep learning development and deployment across different frameworks and platforms. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 810–822. IEEE, 2019.

[116] G. Gupta and N. Mangla. Trust aware multi-objective metaheuristics for workflow scheduling in cloud computing. In *2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON)*, volume 1, pages 602–609. IEEE, 2022.

[117] K. Gyarmathy. The benefits, potential, and future of edge computing. https://www.vxchnges.com/blog/the-5-best-benefits-of-edge-computing/, 2021. Last accessed 2021-02-15.

[118] M. Haakman, L. Cruz, H. Huijgens, and A. van Deursen. Ai lifecycle models need to be revised. *Empirical Software Engineering*, 26(5):1–29, 2021.

[119] N. R. Haddaway, M. J. Page, C. C. Pritchard, and L. A. McGuinness. Prisma2020: An r package and shiny app for producing prisma 2020-compliant flow diagrams, with interactivity for optimised digital transparency and open synthesis. *Campbell Systematic Reviews*, 18(2):e1230, 2022.

[120] P. Hane. Ipwhois. https://pypi.org/project/ipwhois/, 2020. Last accessed 14 March 2022.

[121] S. Harrer. Attention is not all you need: the complicated case of ethically using large language models in healthcare and medicine. *EBioMedicine*, 90, 2023.

[122] HashiCorp. Terraform. https://www.terraform.io/, 2022. Last accessed 14 March 2022.

[123] HashiCorp. Vagrant. https://www.vagrantup.com/, 2022. Last accessed 14 March 2022.

[124] R. Hat. Ansible. https://www.ansible.com/, 2022. Last accessed 14 March 2022.

[125] R. Hat. Red hat openshift. https://www.redhat.com/en/technologies/cloud-computing/openshift, 2022. Last accessed 9 March 2022.

[126] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama. Intelligent service deployment policy for next-generation industrial edge networks. *IEEE Transactions on Network Science and Engineering*, 2021.

[127] J. Hermann and M. D. Balso. Michelangelo. https://eng.uber.com/michelangelo-machine-learning-platform/, 2017. Last accessed 14 March 2022.

[128] L.-A. Hîrţan, C. Dobre, and H. González-Vélez. Blockchain-based reputation for intelligent transportation systems. *Sensors*, 20(3):791, 2020.

[129] M. Hoffmann, L. Malburg, and R. Bergmann. Progan: Toward a framework for process monitoring and flexibility by change via generative adversarial networks. In *International Conference on Business Process Management*, pages 43–55. Springer, 2021.

[130] M. K. Hossain and Q. Meng. A fine-scale spatial analytics of the assessment and mapping of buildings and population at different risk levels of urban flood. *Land Use Policy*, 99:104829, 2020.

[131] M. Hosseini Shirvani, A. M. Rahmani, and A. Sahafi. An iterative mathematical decision model for cloud migration: A cost and security risk approach. *Software: Practice and Experience*, 48(3):449–485, 2018.

[132] Y. Huang, K. cai, R. Zong, and Y. Mao. Design and implementation of an edge computing platform architecture using docker and kubernetes for machine learning. In *Proceedings of the 3rd International Conference on High Performance Compilation, Computing and Communications*, pages 29–32, 2019.

[133] W. Hummer, V. Muthusamy, T. Rausch, P. Dube, K. El Maghraoui, A. Murthi, and P. Oum. Modelops: Cloud-based lifecycle management for reliable and trusted ai. In *2019 IEEE International Conference on Cloud Engineering (IC2E)*, pages 113–120. IEEE, 2019.

[134] S. Hykes. Docker swarm engine. https://docs.docker.com/engine/swarm/.

[135] IBM. Watson studio. https://www.ibm.com/cloud/watson-studio, 2022. Last accessed 13 March 2022.

[136] IBM. Aiops solutions, 2023. Last accessed 29 August 2023.

[137] S. Idowu, D. Strüber, and T. Berger. Asset management in machine learning: a survey. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 51–60. IEEE, 2021.

[138] ieee. Ieee xplore api portal, 2022. Last accessed 19 June 2022.

[139] A. Ignatov. Ai benchmark. https://pypi.org/project/ai-benchmark/, 2021. Last accessed 7 March 2022.

[140] A. Ignatov, R. Timofte, W. Chou, K. Wang, M. Wu, T. Hartley, and L. Van Gool. Ai benchmark: Running deep neural networks on android smartphones. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[141] A. Ignatov, R. Timofte, A. Kulik, S. Yang, K. Wang, F. Baum, M. Wu, L. Xu, and L. Van Gool. Ai benchmark: All about deep learning on smartphones in 2019. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3617–3635. IEEE, 2019.

[142] A. Inc. Core ml, 2022. Last accessed 2 October 2022.

[143] D. Inc. Docker registry. https://docs.docker.com/registry/, 2022. Last accessed 9 March 2022.

[144] I. Inc. Telegraf open source server agent. https://www.influxdata.com/time-series-platform/telegraf/, 2022. Last accessed 8 March 2022.

[145] P. Inc. Pachyderm. https://www.pachyderm.com/, 2022. Last accessed 14 March 2022.

[146] iterative.ai. Open-source version control system for machine learning projects. https://dvc.org/, 2022. Last accessed 14 March 2022.

[147] R. Jabbari, N. bin Ali, K. Petersen, and B. Tanveer. What is devops? a systematic mapping study on definitions and practices. In *Proceedings of the Scientific Workshop Proceedings of XP2016*, pages 1–11, 2016.

[148] H. Jain and K. Deb. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach. *IEEE Transactions on evolutionary computation*, 18(4):602–622, 2013.

[149] N. Janbi, I. Katib, A. Albeshri, and R. Mehmood. Distributed artificial intelligence-as-a-service (daiaas) for smarter ioe and 6g environments. *Sensors*, 20(20):5796, 2020.

[150] M. Janssen, P. Brous, E. Estevez, L. S. Barbosa, and T. Janowski. Data governance: Organizing data for trustworthy artificial intelligence. *Government Information Quarterly*, 37(3):101493, 2020.

[151] L. Jia, Z. Zhou, F. Xu, and H. Jin. Cost-efficient continuous edge learning for artificial intelligence of things. *IEEE Internet of Things Journal*, 9(10):7325–7337, 2021.

[152] R. Jin, P. Muench, V. Deenadhayalan, and B. Hatfield. Aiops essential to unified resiliency management in data lakehouses. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 4777–4781. IEEE, 2022.

[153] A. Jindal, S. Qiao, R. Sen, and H. Patel. Microlearner: A fine-grained learning optimizer for big data workloads at microsoft. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2423–2434. IEEE, 2021.

[154] M. M. John, H. H. Olsson, and J. Bosch. Towards mlops: A framework and maturity model. In *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 1–8. IEEE, 2021.

[155] M. M. John, H. H. Olsson, and J. Bosch. Towards an ai-driven business development framework: A multi-case study. *Journal of Software: Evolution and Process*, page e2432, 2022.

[156] Json online validator and formatter - json lint. https://jsonlint.com/. Last accessed 20 September 2023.

[157] G. A. Kaissis, M. R. Makowski, D. Rückert, and R. F. Braren. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, 2(6):305–311, 2020.

[158] I. Karamitsos, S. Albarhami, and C. Apostolopoulos. Applying devops practices of continuous automation for machine learning. *Information*, 11(7):363, 2020.

[159] B. Karlaš, M. Interlandi, C. Renggli, W. Wu, C. Zhang, D. Mukunthu Iyappan Babu, J. Edwards, C. Lauren, A. Xu, and M. Weimer. Building continuous integration services for machine learning. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2407–2415, 2020.

[160] S. Keele et al. Guidelines for performing systematic literature reviews in software engineering. Technical report, Technical report, ver. 2.3 ebse technical report. ebse, 2007.

[161] Z. Khan, A. Anjum, and S. L. Kiani. Cloud based big data analytics for smart future cities. In *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, pages 381–386. IEEE, 2013.

[162] Z. Khan, A. Anjum, K. Soomro, and M. A. Tahir. Towards cloud based big data analytics for smart future cities. *Journal of Cloud Computing*, 4(1), 2015.

[163] P. Kiss, A. Reale, C. J. Ferrari, and Z. Istenes. Deployment of iot applications on 5g edge. In *2018 IEEE International Conference on Future IoT Technologies (Future IoT)*, pages 1–9. IEEE, 2018.

[164] S. Kong, J. Ai, and M. Lu. Cl-mmad: A contrastive learning based multimodal software runtime anomaly detection method. *Applied Sciences*, 13(6):3596, 2023.

[165] G. P. Koudouridis, Q. He, and G. Dán. An architecture and performance evaluation framework for artificial intelligence solutions in beyond 5g radio access networks. *EURASIP Journal on Wireless Communications and Networking*, 2022(1):1–32, 2022.

[166] D. Kreuzberger, N. Kühl, and S. Hirschl. Machine learning operations (mlops): Overview, definition, and architecture. *IEEE Access*, 2023.

[167] F. Kumeno. Sofware engneering challenges for machine learning applications: A literature review. *Intelligent Decision Technologies*, 13(4):463–476, 2019.

[168] A. Lavalle, M. A. Teruel, A. Maté, and J. Trujillo. Improving sustainability of smart cities through visualization techniques for big data from iot devices. *Sustainability*, 12(14):5595, 2020.

[169] M. Lee, X. She, B. Chakraborty, S. Dash, B. Mudassar, and S. Mukhopadhyay. Reliable edge intelligence in unreliable environment. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 896–901. IEEE, 2021.

[170] Y. Lee, A. Scolari, B.-G. Chun, M. Weimer, and M. Interlandi. From the edge to the cloud: Model serving in ML.NET. *IEEE Data Eng. Bull.*, 41(4):46–53, 2018.

[171] S. Leroux, P. Simoens, M. Lootus, K. Thakore, and A. Sharma. Tinymlops: Operational challenges for widespread edge ai adoption. In *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1003–1010. IEEE, 2022.

[172] P. Li, J. Thomas, X. Wang, A. Khalil, A. Ahmad, R. Inacio, S. Kapoor, A. Parekh, A. Doufexi, A. Shojaeifard, et al. Rlops: Development life-cycle of reinforcement learning aided open ran. *IEEE Access*, 2022.

[173] C. Lim, K.-J. Kim, and P. P. Maglio. Smart cities with big data: Reference models, challenges, and considerations. *Cities*, 82:86–99, 2018.

[174] A. Lima, L. Monteiro, and A. P. Furtado. Mlops: Practices, maturity models, roles, tools, and challenges-a systematic literature review. *ICEIS (1)*, pages 308–320, 2022.

[175] M. A. Lones. How to avoid machine learning pitfalls: a guide for academic researchers. *arXiv preprint arXiv:2108.02497*, 2021.

[176] D. G. Lopes. tcp-latency. https://pypi.org/project/tcp-latency/, 2022. Last accessed 25 July 2022.

[177] U. Lopez-Novoa, J. Morgan, K. Jones, O. Rana, T. Edwards, and F. Grigoletto. Enabling citizen science in rural environments with iot and mobile technologies. In *CPSS@ IOT*, pages 50–56, 2019.

[178] J. Ltd. Artifactory - universal artifact repository manager. https://jfrog.com/artifactory/, 2022. Last accessed 9 March 2022.

[179] L. E. Lwakatare, I. Crnkovic, and J. Bosch. Devops for ai–challenges in development of ai-enabled applications. In *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–6. IEEE, 2020.

[180] L. E. Lwakatare, A. Raj, I. Crnkovic, J. Bosch, and H. H. Olsson. Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions. *Information and software technology*, 127:106368, 2020.

[181] W. Ma, R. Wang, Y. Gu, Q. Meng, H. Huang, S. Deng, and Y. Wu. Multi-objective microservice deployment optimization via a knowledge-driven evolutionary algorithm. *Complex & Intelligent Systems*, pages 1–19, 2020.

[182] R. Mahindru, H. Kumar, and S. Bansal. Log anomaly to resolution: Ai based proactive incident remediation. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1353–1357. IEEE, 2021.

[183] S. Mäkinen, H. Skogström, E. Laaksonen, and T. Mikkonen. Who needs mlops: What data scientists seek to accomplish and how can mlops help? In *2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)*, pages 109–112. IEEE, 2021.

[184] G. O. R. Manual. Gurobi optimization, llc. https://www.gurobi.com/, 2021. Last accessed 13 March 2022.

[185] T. Mboweni, T. Masombuka, and C. Dongmo. A systematic review of machine learning devops. In *2022 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, pages 1–6. IEEE, 2022.

[186] R. McCreadie, J. Soldatos, J. Fuerst, M. F. Argerich, G. Kousiouris, J.-D. Totow, A. C. Nieto, B. Q. Navidad, D. Kyriazis, C. Macdonald, et al. Leveraging data-driven infrastructure management to facilitate aiops for big data applications and operations. In *Technologies and Applications for Big Data Value*, pages 135–158. Springer, 2022.

[187] W. McKinney. pandas. https://pandas.pydata.org, 2020. Last accessed 21 September 2023.

[188] A. Medvedev, P. Fedchenkov, A. Zaslavsky, T. Anagnostopoulos, and S. Khoruzhnikov. Waste management as an iot-enabled service in smart cities. In *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*, pages 104–115. Springer, 2015.

[189] P. Meloni, D. Loi, P. Busia, G. Deriu, A. D. Pimentel, D. Sapra, T. Stefanov, S. Minakova, F. Conti, L. Benini, et al. Optimization and deployment of cnns at the edge: the aloha experience. In *Proceedings of the 16th ACM international conference on computing frontiers*, pages 326–332, 2019.

[190] H. Miao, A. Li, L. S. Davis, and A. Deshpande. Modelhub: Deep learning lifecycle management. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1393–1394. IEEE, 2017.

[191] Microsoft. Azure machine learning. https://azure.microsoft.com/en-us/services/machine-learning/, 2022. Last accessed 14 March 2022.

[192] Microsoft. Aiops, 2023. Last accessed 29 August 2023.

[193] R. Miñón, J. Diaz-de Arcaya, A. I. Torre-Bastida, and P. Hartlieb. Pangea: An mlops tool for automatically generating infrastructure and deploying analytic pipelines in edge, fog and cloud layers. *Sensors*, 22(12):4425, 2022.

[194] R. Miñón, J. Díaz-de Arcaya, A. I. Torre-Bastida, G. Zarate, and A. Moreno-Fernandez-de Leceta. Mlpacker: A unified software tool for packaging and deploying atomic and distributed analytic pipelines. In *2022 7th International Conference on Smart and Sustainable Technologies (SpliTech)*, pages 1–6. IEEE, 2022.

[195] S. Mnasri, N. Nasri, A. Van Den Bossche, and T. Val. A hybrid ant-genetic algorithm to solve a real deployment problem: a case study with experimental validation. In *International Conference on Ad-Hoc Networks and Wireless*, pages 367–381. Springer, 2017.

[196] B. Momjian. *PostgreSQL: introduction and concepts*, volume 192. Addison-Wesley New York, 2001.

[197] R. M. Morais. On the suitability, requisites, and challenges of machine learning. *Journal of Optical Communications and Networking*, 13(1):A1–A12, 2021.

[198] S. Moreschini, F. Pecorelli, X. Li, S. Naz, D. Hästbacka, and D. Taibi. Cloud continuum: the definition. *IEEE Access*, 10:131876–131886, 2022.

[199] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan, et al. Ray: A distributed framework for emerging {AI} applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 561–577, 2018.

[200] A. Munappy, J. Bosch, H. H. Olsson, A. Arpteg, and B. Brinne. Data management challenges for deep learning. In *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 140–147. IEEE, 2019.

[201] A. R. Munappy, D. I. Mattos, J. Bosch, H. H. Olsson, and A. Dakkak. From ad-hoc data analytics to dataops. In *Proceedings of the International Conference on Software and System Processes*, pages 165–174, 2020.

[202] A. Navidi and F. A.-S. Khatami. Energy management and planning in smart cities. *CIRED-Open Access Proceedings Journal*, 2017(1):2723–2725, 2017.

[203] S. M. Neuman, B. Plancher, B. P. Duisterhof, S. Krishnan, C. Banbury, M. Mazumder, S. Prakash, J. Jabbour, A. Faust, G. C. de Croon, et al. Tiny robot learning: Challenges and directions for machine learning in resource-constrained robots. *arXiv preprint arXiv:2205.05748*, 2022.

[204] P. Niemelä, B. Silverajan, M. Nurminen, J. Hukkanen, and H.-M. Järvinen. Laops: Learning analytics with privacy-aware mlops. In *CSEDU (2)*, pages 213–220, 2022.

[205] L. Nikolov and V. Slavyanov. Network infrastructure for cybersecurity analysis. In *International scientific conference*, 2018.

[206] P. Notaro, J. Cardoso, and M. Gerndt. A systematic mapping study in aiops. In *International Conference on Service-Oriented Computing*, pages 110–123. Springer, 2020.

[207] P. Notaro, J. Cardoso, and M. Gerndt. A survey of aiops methods for failure management. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 12(6):1–45, 2021.

[208] P. Notaro, S. Haeri, J. Cardoso, and M. Gerndt. Logrule: Efficient structured log mining for root cause analysis. *IEEE Transactions on Network and Service Management*, 2023.

[209] P.-E. Novac, G. Boukli Hacene, A. Pegatoquet, B. Miramond, and V. Gripon. Quantization and deployment of deep neural networks on microcontrollers. *Sensors*, 21(9):2984, 2021.

[210] A. A. Obinikpo and B. Kantarci. Big sensed data meets deep learning for smarter health care in smart cities. *Journal of Sensor and Actuator Networks*, 6(4):26, 2017.

[211] L. C. Ochei, A. Petrovski, and J. M. Bass. Optimal deployment of components of cloud-hosted application for guaranteeing multitenancy isolation. *Journal of Cloud Computing*, 8(1):1–38, 2019.

[212] M. Olenčin and J. Perháč. Automated configuration of a linux web server security. In *2019 IEEE 15th International Scientific Conference on Informatics*, pages 000491–000496. IEEE, 2019.

[213] T. Oliphant. Numpy. https://numpy.org, 2020. Last accessed 21 September 2023.

[214] C. Olston, N. Fiedel, K. Gorovoy, J. Harmsen, L. Lao, F. Li, V. Rajashekhar, S. Ramesh, and J. Soyke. Tensorflow-serving: Flexible, high-performance ml serving. *arXiv preprint arXiv:1712.06139*, 2017.

[215] OpenFaaS. Openfaas - serverless functions made simple. https://www.openfaas.com/, 2022. Last accessed 10 March 2022.

[216] M. Openja, F. Majidi, F. Khomh, B. Chembakottu, and H. Li. Studying the practices of deploying machine learning projects on docker. *arXiv preprint arXiv:2206.00699*, 2022.

[217] Oracle. Oracle vm virtualbox. https://www.virtualbox.org/, 2022. Last accessed 14 March 2022.

[218] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar. Towards the deployment of machine learning solutions in network traffic classification: A systematic survey. *IEEE Communications Surveys & Tutorials*, 21(2):1988–2014, 2018.

[219] M. J. Page, J. E. McKenzie, P. M. Bossuyt, I. Boutron, T. C. Hoffmann, C. D. Mulrow, L. Shamseer, J. M. Tetzlaff, E. A. Akl, S. E. Brennan, et al. The prisma 2020 statement: an updated guideline for reporting systematic reviews. *International Journal of Surgery*, 88:105906, 2021.

[220] M. J. Page, D. Moher, P. M. Bossuyt, I. Boutron, T. C. Hoffmann, C. D. Mulrow, L. Shamseer, J. M. Tetzlaff, E. A. Akl, S. E. Brennan, et al. Prisma 2020 explanation and elaboration: updated guidance and exemplars for reporting systematic reviews. *Bmj*, 372, 2021.

[221] A. Paleyes, C. Cabrera, and N. D. Lawrence. An empirical evaluation of flow based programming in the machine learning deployment context. *arXiv preprint arXiv:2204.12781*, 2022.

[222] L. Patrono, L. Atzori, P. Šolić, M. Mongiello, and A. Almeida. Challenges to be addressed to realize internet of things solutions for smart environments, 2020.

[223] P. Paudyal, S. Shen, S. Yan, and D. Simeonidou. Toward deployments of ml applications in optical networks. *IEEE Photonics Technology Letters*, 33(11):537–540, 2021.

[224] Y. Peng, C. Liu, S. Liu, Y. Liu, and Y. Wu. Smarttro: Optimizing topology robustness for internet of things via deep reinforcement learning with graph convolutional networks. *Computer Networks*, 218:109385, 2022.

[225] F. Pezoa, J. L. Reutter, F. Suarez, M. Ugarte, and D. Vrgoč. Foundations of json schema. In *Proceedings of the 25th International Conference on World Wide Web*, pages 263–273, 2016.

[226] C. Picardi, C. Paterson, R. D. Hawkins, R. Calinescu, and I. Habli. Assurance argument patterns and processes for machine learning in safety-related systems. In *Proceedings of the Workshop on Artificial Intelligence Safety (SafeAI 2020)*, pages 23–30. CEUR Workshop Proceedings, 2020.

[227] J. Pivarski, C. Bennett, and R. L. Grossman. Deploying analytics with the portable format for analytics (PFA). In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 579–588. ACM, 2016.

[228] P. Platypus. David hadka. https://github.com/Project-Platypus/Platypus, 2021. Last accessed 13 March 2022.

[229] PMML. Data mining group. https://dmg.org/pmml/pmml-v4-4.html, 2009. Last accessed 13 March 2022.

[230] L. Poenaru-Olaru, L. Cruz, J. S. Rellermeyer, and A. Van Deursen. Maintaining and monitoring aiops models against concept drift. In *2023 IEEE/ACM 2nd International Conference on AI Engineering–Software Engineering for AI (CAIN)*, pages 98–99. IEEE, 2023.

[231] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich. Data lifecycle challenges in production machine learning: a survey. *ACM SIGMOD Record*, 47(2):17–28, 2018.

[232] M. Popp. Comprehensive support of the lifecycle of machine learning models in model management systems. Master's thesis, University of Stuttgart, 2019.

[233] J. Prasad, A. Jain, and U. E. Zachariah. Comparative evaluation of machine learning development lifecycle tools. In *2022 International Conference on Recent Trends in Microelectronics, Automation, Computing and Communications Systems (ICMACC)*, pages 1–6. IEEE, 2022.

[234] D. C. Price, E. van der Velden, S. Celles, P. T. Eendebak, M. M. McKerns, E. M. Olson, C. Raffel, B. Yi, and E. Ash. Hickle: A hdf5-based python pickle replacement. *Journal of Open Source Software*, 3(32):1115, 2018.

[235] C. Profentzas, M. Almgren, and O. Landsiedel. Minilearn: On-device learning for low-power iot devices. In *Proceedings of the 2022 International Conference on Embedded Wireless Systems and Networks (Linz, Austria)(EWSN'22). Junction Publishing, USA*, 2022.

[236] F. Project. Fast and simple serverless functions for kubernetes. https://fission.io/, 2022. Last accessed 10 March 2022.

[237] S. Quevedo, F. Merchán, R. Rivadeneira, and F. X. Dominguez. Evaluating apache openwhisk-faas. In *2019 IEEE Fourth Ecuador Technical Chapters Meeting (ETCM)*, pages 1–5. IEEE, 2019.

[238] M. M. Rathore, A. Ahmad, A. Paul, and S. Rho. Urban planning and building smart cities based on the internet of things using big data analytics. *Computer Networks*, 101:63 – 80, 2016. Industrial Technologies and Applications for the Internet of Things.

[239] L. Rijal, R. Colomo-Palacios, and M. Sánchez-Gordón. Aiops: A multivocal literature review. *Artificial Intelligence for Cloud and Edge Computing*, pages 31–50, 2022.

[240] D. Rock. Engineering value: The returns to technological talent and investments in artificial intelligence. *Available at SSRN 3427412*, 2019.

[241] G. S. Rodrigues, F. P. Guimarães, G. N. Rodrigues, A. Knauss, J. P. C. de Araújo, H. Andrade, and R. Ali. Goald: A goal-driven deployment framework for dynamic and heterogeneous computing environments. *Information and software technology*, 111:159–176, 2019.

[242] L. Rothkrantz. Flood control of the smart city prague. In *2016 Smart Cities Symposium Prague (SCSP)*, pages 1–7. IEEE, 2016.

[243] P. Ruf, M. Madan, C. Reich, and D. Ould-Abdeslam. Demystifying mlops and presenting a recipe for the selection of open-source tools. *Applied Sciences*, 11(19):8861, 2021.

[244] T. Saba, A. Rehman, K. Haseeb, S. A. Bahaj, and G. Jeon. Energy-efficient edge optimization embedded system using graph theory with 2-tiered security. *Electronics*, 11(18):2942, 2022.

[245] M. K. Saggi and S. Jain. A survey towards an integration of big data analytics to big insights for value-creation. *Information Processing & Management*, 54(5):758–790, 2018.

[246] R. Sánchez-Corcuera, A. Nuñez-Marcos, J. Sesma-Solance, A. Bilbao-Jayo, R. Mulero, U. Zulaika, G. Azkune, and A. Almeida. Smart cities survey: Technologies, application domains and challenges for the cities of the future. *International Journal of Distributed Sensor Networks*, 15(6):1550147719853984, 2019.

[247] S. Schelter, F. Biessmann, T. Januschowski, D. Salinas, S. Seufert, and G. Szarvas. On challenges in machine learning model management. *IEEE Data Engineering Bulletin*, 2015.

[248] S. Schöbel, A. Schmitt, D. Benner, M. Saqr, A. Janson, and J. M. Leimeister. Charting the evolution and future of conversational agents: A research agenda along five waves and new frontiers. *Information Systems Frontiers*, pages 1–26, 2023.

[249] K. Schwab. *The fourth industrial revolution*. Currency, 2017.

[250] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison. Hidden technical debt in machine learning systems. *Advances in neural information processing systems*, 28, 2015.

[251] S. Sharma and G. P. Rangaiah. Multi-objective optimization applications in chemical engineering. *Multi-Objective Optimization in Chemical Engineering: Developments and Applications*, pages 35–102, 2013.

[252] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[253] R. M. Shukla and J. Cartlidge. Challenges faced by industries and their potential solutions in deploying machine learning applications. In *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0119–0124. IEEE, 2022.

[254] P. Singh. Machine learning deployment using kubernetes. In *Deploy Machine Learning Models to Production*, pages 127–146. Springer, 2021.

[255] D. Sobnath, I. U. Rehman, and M. M. Nasralla. Smart cities to improve mobility and quality of life of the visually impaired. In *Technological Trends in Improved Mobility of the Visually Impaired*, pages 3–28. Springer, 2020.

[256] H. Song, H. Ji, Y. Yu, and B. Xie. A review of observability issues in hospital information system. In *2022 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, pages 1–7. IEEE, 2022.

[257] N. Soni, E. K. Sharma, N. Singh, and A. Kapoor. Artificial intelligence in business: From research and innovation to market deployment. *Procedia Computer Science*, 167:2200–2210, 2020.

[258] R. Sothilingam, V. Pant, and E. Yu. Using i* to analyze collaboration challenges in mlops project teams. In *Proceedings of the 15th International iStar Workshop iStar 2022)*, 2022.

[259] spinnaker. Spinnaker. https://spinnaker.io/, 2022. Last accessed 13 March 2022.

[260] Springer. Springer nature api portal, 2022. Last accessed 18 June 2022.

[261] M. Steidl, M. Felderer, and R. Ramler. The pipeline for the continuous development of artificial intelligence models—current state of research and practice. *Journal of Systems and Software*, page 111615, 2023.

[262] stressng. stress-ng. https://wiki.ubuntu.com/Kernel/Reference/stress-ng, 2020. Last accessed 13 March 2022.

[263] R. Subramanya, S. Sierla, and V. Vyatkin. From devops to mlops: Overview and application to electricity market forecasting. *Applied Sciences*, 12(19):9851, 2022.

[264] Q. Sun, C. Bai, H. Geng, and B. Yu. Deep neural network hardware deployment optimization via advanced active learning. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1510–1515. IEEE, 2021.

[265] G. Symeonidis, E. Nerantzis, A. Kazakis, and G. A. Papakostas. Mlops-definitions, tools and challenges. In *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0453–0460. IEEE, 2022.

[266] D. A. Tamburri. Sustainable mlops: Trends and challenges. In *2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 17–23. IEEE, 2020.

[267] B. Tang, Z. Chen, G. Hefferman, S. Pei, T. Wei, H. He, and Q. Yang. Incorporating intelligence in fog computing for big data analysis in smart cities. *IEEE Transactions on Industrial informatics*, 13(5):2140–2150, 2017.

[268] S. C. K. Tekouabou, W. Cherif, H. Silkan, et al. Improving parking availability prediction in smart cities with iot and ensemble-based model. *Journal of King Saud University-Computer and Information Sciences*, 2020.

[269] TensorFlow. Tensorflow lite, 2022. Last accessed 2 October 2022.

[270] TensorFlow. Tensorflow serving. https://www.tensorflow.org/tfx/guide/serving, 2022. Last accessed 14 March 2022.

[271] The official YAML web site. https://yaml.org/. Last accessed 20 September 2023.

[272] K. Thonglek, K. Takahashi, K. Ichikawa, C. Nakasan, H. Nakada, R. Takano, P. Leelaprute, and H. Iida. Automated quantization and retraining for neural network models without labeled data. *IEEE Access*, 10:73818–73834, 2022.

[273] A. I. Torre-Bastida, J. Diaz-de-Arcaya, R. Miñon, and A. Almeida. Method, system and computer program product for optimizing the deployment of analytical pipelines. European Patent Office EP 4,064,047 A1, 2022. Pending.

[274] Travis ci. https://travis-ci.org/, 2020. Last accessed 20 September 2023.

[275] P.-H. Tsai, H.-J. Hong, A.-C. Cheng, and C.-H. Hsu. Distributed analytics in fog computing platforms using tensorflow and kubernetes. In *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 145–150. IEEE, 2017.

[276] C. University. arxiv api access, 2021. Last accessed 18 June 2022.

[277] M. C. Urdaneta-Ponte, A. Mendez-Zorrilla, and I. Oleagordia-Ruiz. Recommendation systems for education: systematic review. *Electronics*, 10(14):1611, 2021.

[278] M. Usman, S. Ferlin, A. Brunstrom, and J. Taheri. A survey on observability of distributed edge & container-based microservices. *IEEE Access*, 2022.

[279] valgrind. massif. https://valgrind.org/docs/manual/ms-manual.html, 2021. Last accessed 14 March 2022.

[280] Valohai. Valohai | take ml places it's never been. https://valohai.com/, 2023. Last accessed 1 March 2023.

[281] G. Van Rossum and F. L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[282] S. I. Venieris, I. Panopoulos, I. Leontiadis, and I. S. Venieris. How to reach real-time ai on consumer devices? solutions for programmable and custom architectures. In *2021 IEEE 32nd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 93–100. IEEE, 2021.

[283] S. I. Venieris, I. Panopoulos, and I. S. Venieris. Oodin: An optimised on-device inference framework for heterogeneous mobile devices. In *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–8. IEEE, 2021.

[284] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer. A survey on distributed machine learning. *ACM Computing Surveys (CSUR)*, 53(2):1–33, 2020.

[285] E. Verenich, A. Velasquez, M. S. Murshed, and F. Hussain. Flexserve: Deployment of pytorch models as flexible {REST} endpoints. In *2020 {USENIX} Conference on Operational Machine Learning (OpML 20)*, 2020.

[286] M. Vergin Raja Sarobin. Optimized node deployment in wireless sensor network for smart grid application. *Wireless Personal Communications*, 111(3):1431–1451, 2020.

[287] S. Vergis, V. Komianos, G. Tsoumanis, A. Tsipis, and K. Oikonomou. A low-cost vehicular traffic monitoring system using fog computing. *Smart Cities*, 3(1):138–156, 2020.

[288] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes. Large-scale cluster management at Google with Borg. In *Proceedings of the Tenth European Conference on Computer Systems*, pages 1–17, 2015.

[289] B. Vogel-Heuser, E. Trunzer, D. Hujo, and M. Sollfrank. (re) deployment of smart algorithms in cyber–physical production systems using dsl4hdncs. *Proceedings of the IEEE*, 109(4):542–555, 2021.

[290] W. W. W. C. (W3C). Quality dimensions defined in iso/iec 25012. https://www.w3.org/TR/vocab-dqv/#DimensionsOfISOIEC25012. Last accessed 20 September 2023.

[291] S.-L. Wamba-Taguimdje, S. F. Wamba, J. R. K. Kamdjoug, and C. E. T. Wanko. Influence of artificial intelligence (ai) on firm performance: the business value of ai-based transformation projects. *Business Process Management Journal*, 2020.

[292] L. Wang and D. Sng. Deep learning algorithms with applications to video analytics for a smart city: A survey. *arXiv preprint arXiv:1512.03131*, 2015.

[293] P. Wang, J. Huang, Z. Cui, L. Xie, and J. Chen. A gaussian error correction multi-objective positioning model with nsga-ii. *Concurrency and Computation: Practice and Experience*, 32(5):e5464, 2020.

[294] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen. Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(2):869–904, 2020.

[295] Z. Wang, H. Xie, D. He, and S. Chan. Wireless sensor network deployment optimization based on two flower pollination algorithms. *IEEE Access*, 7:180590–180608, 2019.

[296] J. Wen, Z. Chen, Y. Liu, Y. Lou, Y. Ma, G. Huang, X. Jin, and X. Liu. An empirical study on challenges of application development in serverless computing. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 416–428, 2021.

[297] T. Wittkopp, P. Wiesner, D. Scheinert, and A. Acker. Loglab: attention-based labeling of log data anomalies via weak supervision. In *International Conference on Service-Oriented Computing*, pages 700–707. Springer, 2021.

[298] G. Wu, W. Bao, X. Zhu, W. Xiao, and J. Wang. Optimal dynamic reserved bandwidth allocation for cloud-integrated cyber-physical systems. *IEEE Access*, 5:26224–26236, 2017.

[299] C. Xia, J. Zhao, H. Cui, X. Feng, and J. Xue. Dnntune: Automatic benchmarking dnn models for mobile-cloud computing. *ACM Transactions on Architecture and Code Optimization (TACO)*, 16(4):1–26, 2019.

[300] Y. Xiong, Y. Sun, L. Xing, and Y. Huang. Extend cloud to edge with kubeedge. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 373–377. IEEE, 2018.

[301] J. Xu, G. Wang, Y. Yao, Z. Li, C. Cao, H. Tong, et al. A deep learning dataloader with shared data preparation. *Advances in Neural Information Processing Systems*, 35:17146–17156, 2022.

[302] X. Xu, S. Fu, W. Li, F. Dai, H. Gao, and V. Chang. Multi-objective data placement for workflow management in cloud infrastructure using nsga-ii. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(5):605–615, 2020.

[303] Yamllint - the yaml validator. https://www.yamllint.com/. Last accessed 20 September 2023.

[304] S. Yang, K. Xu, L. Cui, Z. Ming, Z. Chen, and Z. Ming. Ebi-pai: Toward an efficient edge-based iot platform for artificial intelligence. *IEEE Internet of Things Journal*, 8(12):9580–9593, 2020.

[305] J. Yin, S. Gahlot, N. Laanait, K. Maheshwari, J. Morrison, S. Dash, and M. Shankar. Strategies to deploy and scale deep learning on the summit supercomputer. In *2019 IEEE/ACM Third Workshop on Deep Learning on Supercomputers (DLS)*, pages 84–94. IEEE, 2019.

[306] M. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S. A. Hong, A. Konwinski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, et al. Accelerating the machine learning lifecycle with mlflow. *IEEE Data Eng. Bull.*, 41(4):39–45, 2018.

[307] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.

[308] S. A. R. Zaidi, A. M. Hayajneh, M. Hafeez, and Q. Ahmed. Unlocking edge intelligence through tiny machine learning (tinyml). *IEEE Access*, 10:100867–100877, 2022.

[309] G. Zárate, R. Miñón, J. Díaz-de Arcaya, and A. I. Torre-Bastida. K2e: Building mlops environments for governing data and models catalogues while tracking versions. In *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*, pages 206–209. IEEE, 2022.

[310] A. A. Zeeshan. Automating everything as code. In *DevSecOps for. NET Core*, pages 109–162. Springer, 2020.

[311] J. M. Zhang, M. Harman, L. Ma, and Y. Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 2020.

[312] L. Zhang, L. Chen, S. Xia, Y. Ge, C. Wang, and H. Feng. Multi-objective optimization for helium-heated reverse water gas shift reactor by using nsga-ii. *International Journal of Heat and Mass Transfer*, 148:119025, 2020.

[313] L. Zhao, W. Tan, L. Xu, N. Xie, and L. Huang. Crowd-based cooperative task allocation via multicriteria optimization and decision-making. *IEEE Systems Journal*, 14(3):3904–3915, 2020.

[314] J. Zhaoxue, L. Tong, Z. Zhenguo, G. Jingguo, Y. Junling, and L. Liangxiong. A survey on log research of aiops: methods and trends. *Mobile Networks and Applications*, 26(6):2353–2364, 2021.

[315] J. Zhaoxue, L. Tong, Z. Zhenguo, G. Jingguo, Y. Junling, and L. Liangxiong. A survey on log research of aiops: Methods and trends. *Mobile Networks and Applications*, pages 1–12, 2022.

[316] B. Zhou, Y. Svetashova, T. Pychynski, I. Baimuratov, A. Soylu, and E. Kharlamov. Semfe: Facilitating ml pipeline development with semantics. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3489–3492, 2020.

[317] J. Zhou, A. Velichkevich, K. Prosvirov, A. Garg, Y. Oshima, and D. Dutta. Katib: A distributed general automl platform on kubernetes. In *2019 {USENIX} Conference on Operational Machine Learning (OpML 19)*, pages 55–57, 2019.

A

Article assessment

Table A.1 Article assessment based on the quality metrics.

Paper	M1	M2	M3	M4	M5	M6	M7	Total
[2]	1	0	1.2	1	0	0	0	3.2
[5]	1	0	0.4	0	0	0	2	3.4
[7]	1	0	1.2	1	1	0	1.3	5.5
[6]	0	0	0.8	1	0	0	0.7	2.5
[10]	0	1	0.8	0	0	0	2	3.8
[16]	1	1	0.8	0	0	0	0	2.8
[18]	1	0	0.4	1	0	0	0.7	3.1
[19]	1	0	0.4	1	0	0	0	2.4
[20]	1	0	0.4	0	0	0	0	1.4
[25]	1	0	0.8	0	0	0	0.7	2.5
[29]	1	0	1.2	1	1	0	1.3	5.5
[32]	0	0	1.6	1	0	0	2	4.6
[31]	0	1	1.2	0	0	0	0.7	2.9

Continued on next page

Table A.1 – *Continued from previous page*

Paper	M1	M2	M3	M4	M5	M6	M7	Total
[34]	1	1	1.2	1	1	0	2	7.2
[38]	0	0	0.4	1	0	0	0	1.4
[40]	0	0	0.8	1	1	0	1.3	4.1
[43]	1	0	0.8	0	0	0	2	3.8
[46]	0	0	0.4	0	0	0	1.3	1.7
[47]	0	0	0.4	0	1	0	0	1.4
[59]	0	1	0.8	1	1	1	1.3	6.1
[60]	0	0	0.8	0	0	0	2	2.8
[66]	0	0	0.4	1	1	0	2	4.4
[68]	0	0	0.4	1	1	0	0.7	3.1
[72]	1	0	0.4	1	0	0	2	4.4
[78]	1	1	0.8	0	0	0	1.3	4.1
[80]	1	1	0.8	0	1	1	0.7	5.5
[87]	1	0	1.2	0	0	0	0	2.2
[92]	0	0	0.4	1	1	0	0.7	3.1
[99]	0	0	1.2	0	0	0	0.7	1.9
[106]	1	0	0.4	0	0	0	0	1.4
[107]	0	0	0.8	1	1	0	2	4.8
[109]	0	1	1.2	1	1	0	0.7	4.9
[110]	0	0	0.8	0	0	0	2	2.8
[114]	0	0	0.4	1	0	0	0	1.4
[115]	1	0	0.8	1	0	0	2	4.8
[116]	1	0	0.4	0	0	0	0	1.4
[118]	1	0	0.8	1	1	1	0.7	5.5
[121]	0	0	1.2	1	0	0	0	2.2
[126]	0	0	0.8	0	1	1	0.7	3.5
[129]	1	0	1.2	0	0	0	0	2.2
[133]	1	1	0.8	0	0	0	2	4.8
[137]	1	0	0.8	1	0	0	0.7	3.5
[149]	1	1	0.8	1	1	0	2	6.8
[151]	0	0	0.8	0	1	1	0	2.8
[152]	0	0	0.4	0	0	0	0	0.4
[153]	0	1	1.2	0	0	0	0.7	2.9

Continued on next page

Table A.1 – *Continued from previous page*

Paper	M1	M2	M3	M4	M5	M6	M7	Total
[155]	1	1	0.8	1	1	0	0	4.8
[163]	1	0	1.2	0	0	0	2	4.2
[164]	0	1	0.8	1	1	0	0	3.8
[165]	0	0	0.8	1	1	0	0	2.8
[169]	0	0	0.8	0	0	0	0.7	1.5
[171]	0	0	0.8	1	0	0	0	1.8
[179]	1	0	0.4	0	0	0	0.7	2.1
[180]	1	0	1.2	0	1	0	2	5.2
[182]	0	1	1.2	0	0	0	1.3	3.5
[183]	1	0	0.4	1	0	0	0.7	3.1
[189]	0	0	1.2	1	0	0	1.3	3.5
[193]	1	0	0.4	1	1	0	0	3.4
[197]	1	0	0.8	0	1	1	1.3	5.1
[203]	0	0	0.4	1	0	0	0	1.4
[206]	1	0	1.2	1	0	0	0.7	3.9
[208]	0	0	0.8	0	1	0	0	1.8
[209]	1	0	1.2	1	1	0	1.3	5.5
[211]	0	0	0.8	1	1	0	1.3	4.1
[216]	1	0	0.8	1	0	0	0.7	3.5
[218]	1	0	0.8	1	1	1	2	6.8
[221]	0	0	0.4	1	0	0	0	1.4
[223]	0	0	1.2	1	1	0	0.7	3.9
[224]	0	1	0.4	0	1	1	0	3.4
[230]	0	0	1.2	0	0	0	0	1.2
[231]	1	0	0.4	0	1	0	2	4.4
[235]	1	0	0.4	0	0	0	0	1.4
[239]	1	1	0.8	0	0	0	0	2.8
[244]	1	0	0.4	1	1	0	0	3.4
[248]	1	0	0.4	1	1	1	0	0.4
[253]	0	0	0.4	1	0	0	0	1.4
[256]	1	0	0.4	0	0	0	0	1.4
[258]	0	0	0.4	0	0	0	0	0.4
[264]	0	0	0.8	0	0	0	0.7	1.5

Continued on next page

Table A.1 – *Continued from previous page*

Paper	M1	M2	M3	M4	M5	M6	M7	Total
[265]	0	0	0.8	1	0	0	0	1.8
[266]	1	0	0.8	0	0	0	1.3	3.1
[272]	0	1	1.2	1	1	0	0	4.2
[283]	0	0	1.2	1	0	0	0.7	2.9
[282]	0	0	0.8	1	0	0	0.7	2.5
[286]	1	0	1.2	0	1	0	1.3	4.5
[289]	1	0	1.2	1	1	1	1.3	6.5
[296]	1	0	0.8	0	0	0	0.7	2.5
[297]	0	0	1.2	1	0	0	0.7	2.9
[304]	1	0	1.6	0	1	1	0.7	5.3
[305]	0	0	0.4	0	0	0	2	2.4
[308]	0	0	0.8	1	1	0	0	2.8
[309]	1	0	0.4	0	0	0	0	1.4
[316]	1	1	0.8	0	0	0	2	4.8