Architectural Tactics to Achieve Quality Attributes of Machine-Learning-Enabled Systems: A Systematic Literature Review

Vladislav Indykov[a,*], Daniel Strüber[a,b], Rebekka Wohlrab[a,c]

[a] *University of Gothenburg and Chalmers University of Technology, Gothenburg, Sweden*
[b] *Radboud University, Nijmegen, Netherlands*
[c] *Carnegie Mellon University, Pittsburgh, USA*

Abstract

Machine-learning-enabled systems are becoming increasingly common in different industries. Due to the impact of uncertainty and the pronounced role of data, ensuring the quality of such systems requires consideration of several unique characteristics in addition to traditional ones. This range of quality attributes can be achieved by the implementation of specific architectural tactics. Such architectural decisions affect the further functioning of the system and its compliance with business goals. Architectural decisions have to be made with attention to possible quality trade-offs to prevent the cost of mitigating unintended side effects. A related work analysis revealed the need for a thorough study of existing architectural decisions and their impact on various quality attributes in the context of machine-learning-enabled systems. In this paper, to address this goal, we present comprehensive research on the quality of such systems, architectural tactics, and their possible quality consequences. Based on a systematic literature review of 206 primary sources, we identified 11 common quality attributes, and 16 relevant architectural tactics together along with 85 potential quality trade-offs. Our results systematize existing research in building architectures of ML-enabled systems. They can be used by software architects and researchers at the system design stage to estimate the possible consequences of decisions made.

*Corresponding author
Email address: indykov@chalmers.se (Vladislav Indykov)
URL: https://euphort.se/ (Vladislav Indykov)

Author preprint of a paper to appear in *Journal of Systems and Software*

Keywords: machine learning, software architecture, software quality, quality attributes.

1. Introduction

Machine-learning-enabled (ML-enabled) systems [1] are currently in high demand among various spheres. The design, development, and implementation of such systems are widespread now since ML technologies allow organizations to reach results that are difficult to achieve through traditional solutions. Machine learning systems typically work with large volumes of data, adapt, learn, search for, and process complex correlations. The development of AI-based systems is an extremely relevant strategy for the world's largest vendors: Meta is implementing ML components for content moderation and feed personalization, Microsoft is focused on developing the AI companion called "Microsoft Copilot", the use of large language models is conquering new frontiers. However, designing such systems remains a non-trivial and non-standardized task due to the lack of detailed system-level guidelines and instructions for constructing appropriate architectures with a consideration of system specifics.

The construction of ML-based software architecture starts with the collection of requirements, particularly, non-functional ones, also known as *quality attributes*. They must be considered at the stage of architectural design to align the system with the intended goals. As Monson stated: *"You don't drive the architecture, the requirements do. You do your best to serve their needs*" [2].

There are several detailed specifications and standards (e.g., ISO/IEC 25010 [3] and ISO/IEC 45010 [4]) for traditional software that makes it possible to predetermine the fundamental quality attributes of the designed system without conducting any deep research. Some of their quality characteristics can be adapted, updated, and extended to directly meet the needs of ML-based software due to its unique characteristics compared to traditional ones. Specifically, ML-enabled systems operate in environments of high uncertainty and depend on the quality and quantity of data used for model training, validation, and testing. This fact and its relevance were confirmed by the ISO/IEC 25059 [5] issued in June 2023, which adjusted some of the existing qualities from ISO/IEC 25010 to ML contexts and additionally considered several ML-specific aspects (e.g., ethics, transparency). While

this new standard presents an important initiative for addressing the specific quality aspects of ML-enabled systems, it has not been investigated to which extent it characterizes the relevant aspects of this domain exhaustively. Such an investigation could be supported by a systematic study, as we perform in this work.

Quality attributes can be achieved by architectural and non-architectural tactics. Non-architectural ways of achieving quality are based on organizational non-technical management and on technical decisions that do not affect software architecture. Such decisions are too dependent on the system specifics and are out of our scope. Architectural tactics, on the contrary, are general design decisions. They are designed to improve one specific target quality attribute. In practice, it is very common that architectural tactics entail unanticipated tradeoffs on other quality attributes. To raise awareness of the consequences that come with a selection of architectural tactics, it is important to make these tradeoffs explicit. This is one of the contributions of this paper and will enable architects to more deliberately select architectural tactics for ML-enabled systems in the future.

For example, there is an architectural tactic to implement a real-time data monitoring module. In the context of ML-enabled systems, by implementing this architectural tactic, the operator gets an opportunity to monitor all the data used for model training, testing, and validation as well as dynamic input data. Such a decision can increase fairness, reliability, maintainability, accuracy, and security. However, the main trade-off after the implementation of this tactic appears in terms of resource efficiency when operating with big data [6], [7], [8].

In this paper, we give a comprehensive picture of architectural tactics for the engineering of ML-enabled systems along with quality attributes affected by them. We report on the results of a systematic literature review, in which we extracted information from 206 primary sources about these aspects. Specifically, we made the following contributions:

1. We propose a quality model for ML-enabled systems, focused on the most commonly reported quality attributes in the literature, and compare it to the relevant standards of ISO/IEC 25010 [3] and ISO/IEC 25059 [5].
2. We present a range of architectural tactics that can help achieve identified common quality attributes.

3. We present an analysis of the quality trade-offs of the identified architectural tactics, summarized as an impact matrix.

This paper is accompanied by a supplementary artifact[1] which contains search queries and data extraction sheets.

The rest of this paper is structured as follows: Section 2 discusses related work and introduces the used terminology. Section 3 describes our research methodology. Section 4 presents our results, including the identification and analysis of quality attributes, architectural tactics, and quality trade-offs. Section 5 discusses implications concerning specific attributes, other quality standards, and threats to validity. Section 6 concludes and outlines future work. Section 7 provides a data availability statement. Section 8 presents the acknowledgments.

2. Background

2.1. Context

A *quality attribute (QA)* is a measurable or testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders [9]. In ISO/IEC 9126-1:2001, quality attributes are described as a *"checklist to determine software quality"* [10]. According to Lundberg et al. [11], the quality attributes should guide the design of the software architecture. While stakeholders, usage contexts and, therefore, relevant quality attributes differ from one system to another, one can identify the most widespread quality attributes applied to systems of different natures. In the context of this work, we call them the *"common quality attributes" (CQAs)*.

Quality attributes are related to the term "architecturally significant requirement(s)". However, the latter is entirely specialized to a particular system, based on the needs of certain stakeholders, technical capabilities, internal regulations, etc. "*In gathering [architecturally significant requirements], we should be mindful of the business goals of the organization*" [9]. In this paper, we seek to generalize existing experience, putting the specifics of individual systems aside.

An *architectural tactic (AT)* is a "*technique an architect can use to achieve the required quality attributes*" [9]. By definition, the connection between tactic and certain *quality attributes* is implied. However, our study goes

[1]Supplementary Artifact: https://figshare.com/s/57b4fa3f53caecd4a5b1

further and analyzes the impact of its influence on all identified common quality attributes. Balance or compromises between them are called *quality trade-offs*.

2.2. Related work

The study of software quality for ML-enabled systems is an in-demand topic among researchers and practitioners [12], [13]. Despite the relatively small number of studies published at the time of writing the current paper, a steady positive trend in this domain was noted. The space for interpreting the quality of AI systems has only been partially explored and a conclusive view is yet to form, which is proved by the emergence of different quality models based on industrial experience [14], [15], [16]. Such studies work with non-functional requirements relevant to a certain system and most often receive them from domain experts. The generalizability of such models can be debatable due to context dependence. Their systematization and the identification of the most common quality attributes is a way to build a more generalized picture based on real examples. Such a strategy supports a collection of the most recent materials and makes current research more independent from external inputs.

There are also plenty of review papers on architectural issues in the context of AI-based systems [17], [18], [19]. These papers explore a collection of existing architectural design decisions without a clear reference to system qualities or with a focus on the impact of decisions on individual quality attributes and their metrics in isolation from the overall quality picture of the system. As a result, possible trade-offs often remain unnoticed. In contrast with such studies, we strive to investigate the effects of architectural tactics (ATs) on all the identified quality attributes to provide insights at the architectural level.

3. Methodology

The methodology of *systematic literature review (SLR)* allowed us to work with a large amount of scientific information, find common approaches to different systems, and effectively extract information from different sources. Such opportunities suit the research in the chosen domain. We decided to perform an SLR according to Kitchenham's guidelines [20] as we found them most detailed and highly applicable to the current study of software architectures.

3.1. Review Questions

To achieve the research objectives, three fundamental review questions (RQs) were identified.

RQ1: What are the most frequently reported quality attributes for ML-enabled systems? This question aims to identify the most often emphasized QAs in scientific literature.

RQ2: What architectural tactics have been reported to be effective for ML-enabled systems? This question aims to identify ATs to achieve quality attributes defined in RQ1. If the quality attribute can not be satisfied by any AT, then it is out of scope for RQ2 and RQ3.

RQ3: For each architectural tactic, what is the reported impact on all the identified quality attributes? This question aims to identify quality trade-offs when ATs defined in RQ2 are implemented.

3.2. Inclusion and Exclusion Criteria

Only scientific literature was analyzed in this work, leaving grey literature outside the scope of this study. We used the following *inclusion criteria*:

1. Research scientific papers containing lists of QAs for specific or general ML-enabled system(s);
2. Research and review scientific papers with the description of ATs and their influence on the QA(s) of specific or general ML-enabled system(s).

We used the following *exclusion criteria*:

1. Grey literature;
2. Scientific papers about QAs of non-ML-enabled systems;
3. Scientific papers about ATs in non-ML-enabled systems;
4. Scientific papers about applying ML to address software quality concerns of non-ML-enabled systems;
5. Scientific papers about applying ML to address architectural concerns of non-ML-enabled systems;
6. Exclusively for RQ1: secondary research (literature reviews).

For our investigation of RQ1, in which we counted the number of occurrences of specific quality attributes in the literature, we deliberately excluded secondary studies. This is to avoid bias that would arise if the same primary study and its contained quality attributes are considered several times:

through considered secondary studies and through our own data collection. For RQ2 and RQ3 we found it reasonable to leave secondary research included to expand the search and collect architectural tactics as much as possible. The limitation on grey literature is justified by the availability of a sufficient amount of "white" literature for the current study.

3.3. Data Sources

Our search procedure was targeted to enable precise investigation of the identified research questions. To this end, we selected appropriate digital libraries and determined a suitable publication time frame.

Literature databases. To build up a high-quality review, only publications from journals and conference proceedings indexed by at least one globally significant citation database (e.g., Scopus, Web of Science, etc.) were analyzed. The five most popular and largest online digital libraries were the sources for this research: IEEE Xplore (ieeexplore.ieee.org), ACM Digital Library (dl.acm.org), Springer (springerlink.com), Elsevier (sciencedirect.com), Wiley (onlinelibrary.wiley.com).

Time frame. Since this study seeks to explore the most relevant experience in the field of ML-enabled system design, we decided to limit the number of papers with the earliest date of publication of 2011. This decision was made also in connection with the release of the most recent version of the ISO/IEC 25010, which dates to 2011 [3]. This standard is important for the study since this research seeks, in some sense, to clarify the list of quality attributes from it with a consideration of the ML-enabled specifics and recent research experience. Thus, this review is based on the papers from 2011 to 2024 (the year of writing).

3.4. Data Collection

Each review question implies its own objective. The architectural tactics are often not mentioned in works related to software quality and the trade-offs are often not considered in the works on a certain architectural tactic. Thus, we slightly moved away from the standard approach to a systematic literature review with only one query for all review questions and divided our search strategy into three queries, each of which corresponded to its own RQ.

The research under RQ1 works with a set of scientific papers that contains a list of QAs specific to ML-enabled system(s). In the literature, they can be represented explicitly as a list (e.g., a study of Habibullah et al. [21])

or addressed when describing a certain problem or proposing a solution on a system level (e.g., a study of Vojivr et al. [22]). Preliminary research has shown that in the literature on deep learning systems, neural networks, or artificial intelligence systems, the term "machine learning" may not be explicitly stated in the text of the work. Therefore, we decided to expand the query with the above terms to cover a larger number of papers. The introduction of other ML-related terms (such as "MLOps", "ML Engineering" etc.) could potentially shift focus from architectural scope to a more operational one, while the introduction of other software engineering terms (such as "software quality") could exclude certain papers that did not explicitly mention them. Therefore, we decided not to include those keywords. The resulting query for RQ1 is presented below:

```
("machine learning" OR "deep learning" OR "artificial
intelligence" OR "neural network" OR "AI" OR "ML" OR "DL")
AND ("system" OR software") AND ("quality attribute*" OR
"quality characteristic*" OR "non-functional requirement*"
OR "nonfunctional requirement*" OR "quality model" OR "quality
requirement*")
```

Answering RQ2 identifies architectural tactics that improve certain quality attributes. We used search queries based on the results obtained from RQ1, which included common quality attributes (for example, security), together with their sub-characteristics (respectively, privacy). The difficulty of this task is that relevant tactics are not easily identified, since developers might introduce an architectural tactic without referring to it as such. To address this challenge we also included the terms "design pattern" and "architectural decision" in the query. However, we still consider this challenge as a threat to validity and can not argue that the list of collected architectural tactics is complete. Search queries for RQ2 were built according to the template presented below with changing parameters of quality attributes together with their sub-characteristics:

The resulting query for RQ1 is presented below:

```
("machine learning" OR "deep learning" OR "artificial
intelligence" OR "neural network" OR "AI" OR "ML" OR "DL")
AND (system" OR "software") AND ("common quality attribute" OR
"subcharacteristic[1]" OR...  OR "subcharacteristic[n]")
AND ("*architectur* tactic*" OR "design pattern*" OR
"*architectur* design decision*" OR "*architectur*
decision*")
```

The research under RQ3 implies the study of all possible impacts (predominantly positive, predominantly negative, or ambivalent) of the identified architectural tactics from RQ2 on the common quality attributes identified in RQ1. For RQ3 we wrote 16 queries (equal to the number of identified architectural tactics). We expected that the connections between some ATs and some QAs would not be addressed, however, the papers that brought some insights are of special usefulness for the current research. The structure of the search queries corresponds to the template presented below and includes all of the studied common quality attributes and their sub-characteristics together with a changing parameter of architectural tactic:

```
("machine learning" OR "deep learning" OR "artificial
intelligence" OR "neural network" OR "AI" OR "ML" OR "DL")
AND ("common quality attribute[1]" OR ...  OR "common
quality attribute[n]" OR "subcharacteristic[1]" OR...  OR
"subcharacteristic[m]") AND ("architectural tactic[i]") AND
("trade-off*" OR "trade off*" OR "tradeoff*" OR "compromise*")
```

We executed the queries sequentially. The results of data extraction from the sources found with the RQ1-query became the input data for the RQ2-queries, the results of which, similarly, became the input for the RQ3-queries. The full search queries for RQ1, RQ2, and RQ3 as well as the process of data collection are presented in the supplementary artifact[1].

Overall, applying the search procedure with the described queries as well as exclusion and inclusion criteria led to the identification of 206 papers, 37 of which were studied under RQ1, 73 were under RQ2, 96 were under RQ3, and 7 were found for RQ2 but were also found for RQ3 and used to address

[1]Supplementary Artifact: https://figshare.com/s/57b4fa3f53caecd4a5b1

it.

3.5. Data Synthesis

We now discuss the dedicated data synthesis strategies used for each research question as well as our measures taken for ensuring consistency of the data synthesis process.

RQ1. The coding strategy for RQ1 is based on content analysis [23] together with basic frequency analysis and taxonomic analysis [24]. First, we employed content analysis to scrutinize the full-text papers to identify possible quality attributes relevant to ML-enabled systems. In the context of our research, content analysis is a manual research method that examines full texts and concepts of scientific papers, allowing us to comprehensively detect relevant quality attributes across the studies. In order to extract a certain characteristic mentioned in a paper as a quality attribute, we introduced two main conditions: "*the characteristic must be explicitly mentioned in the paper*" and "*the characteristic must describe the quality of the overall system*" (not a certain algorithm or component).

In parallel, we detected that the number of identified attributes was going to be quite large, however, some of them were mentioned only in a few papers. This fact introduces a threat to the generalizability of our findings since such attributes can potentially describe the specifics of only one specific system. To avoid this threat, we made a scoping decision based on the hypothesis: *The more often an attribute is mentioned in different independent papers, the more cases it covers, and therefore the more generalizable it is.* To count those mentions we employed a basic frequency analysis. Our basic frequency analysis can be considered as a form of coding, where the code of a quality attribute is defined as the number of papers mentioning it. It is worth noting that all the papers had equal weight when extracting attributes. One quality attribute could be mentioned explicitly either once or several times in the text of the one paper, however, it did not affect the calculated frequency. This algorithm was applied to all papers found, resulting in a ranked list of quality attributes. Based on the resulting counters, a dividing line was drawn between the frequently mentioned and less frequently mentioned quality attributes. The latter were not included in the common quality model.

We noticed that several frequently mentioned attributes were semantically closely related (e.g., reliability and trustworthiness) or by definition

can be deemed a superset of several other quality attributes (e.g., maintainability usually covered concerns connected to testability, transparency, and maintainability itself). This observation motivated us to employ taxonomic analysis and group quality attributes by semantic similarity to structure the resulting quality model. First, we formulated high-level definitions that discarded the specifics of individual papers, while retaining the fundamental meanings of attributes. Where it was possible, we directly referred to ISO standards ([3],[5]). In other cases we analyzed extra literature to build proper definitions of found attributes. In the studied papers the definitions of quality attributes usually were not mentioned explicitly. Therefore, we analyzed the selected articles again and checked whether our definitions corresponded to the attribute meanings that were implied in them and whether they were relevant in the context of these papers. When the definitions were formulated in a way that satisfied all the cases, we systemized them. We distinguished quality attributes of two levels based on a principle: *"If one quality attribute covers related concerns with certain other attributes and by definition is broader than them, then such an attribute was considered a ("top-level") common quality attribute, while the other associated attributes were deemed "sub-characteristics"*. We note that during the research under subsequent RQs, both common quality attributes and their sub-characteristics are included in search queries. Therefore, the main goal of the taxonomic analysis was to build a clearer perception of the resulting quality model, which is presented graphically as a two-level diagram.

RQ2. Data synthesis and coding strategies for RQ2 were based exclusively on content analysis. Our goal was to explore all relevant ATs we could find with our search strategy for the scope of common quality attributed as determined in RQ1. Therefore, we did not introduce frequency analysis or taxonomic analysis for RQ2. We thoroughly analyzed full-text papers and followed three conditions for extracting data as ATs: the decision must be explicitly mentioned in a paper, the decision must be architectural in nature (it has an impact on the architectural design principle or can be implemented as a part of the overall system architecture) and the decision must be used to improve some quality attribute(s). Those conditions were introduced with a direct connection to the definition of AT used in this research (see Section 2). If an AT is described as effective in achieving multiple quality attributes, it is associated with all affected attributes. To increase generalizability and eliminate bias, the degree of "significance" of an architectural tactic for a

particular attribute was out of scope. For example, if the literature found for RQ2 confirms that the architectural tactic of "containerization" significantly improves both maintainability and portability, then the tactic will be assigned to both attributes, without investigation of which indicator is improved more significantly.

We noticed that some collected tactics only affect the *training system* (e.g., federated learning is usually referred to as a way of organizing model training exclusively), while others can additionally affect other parts of the *deployed system* (e.g., componentization can be the approach to overall system design or be used only to break down the ML pipeline or even the model into components) or be applied to the model when the system is already deployed (e.g., automated bias mitigation usually monitor the outputs of model when it operates with certain inputs). In this context, the *training system* is a system associated with the ML pipeline, which operates with data for model training, testing, and verification; while *deployed system* is a produced ML-enabled system that operates with certain inputs (e.g real-time data).

ML-enabled systems may include the training system into the overall architecture to introduce continuous retraining and improvement based on new data [25]. However, in some cases, the training system can be relatively independent. Therefore, we decided to introduce a classification of the identified tactics depending on which system they affect: training or deployed. Our findings were presented in tabular format.

It is important to note that the results obtained to some extent generalize the experience described in the literature, which means if a tactic was described as effective for at least one type of ML-enabled system (for example, an IoT system), it was included in the table. Consequently, we cannot guarantee with full certainty optimal efficiency for other types of machine learning systems, which is also considered in the analysis of threats to validity.

RQ3. For RQ3, we employed content analysis to identify trade-offs that indicate the impact of implementing architectural tactics on quality attributes. A full-text analysis of the papers identified through our search strategy was performed. We reported an impact of a tactic on a quality attribute if at least one source indicated that applying the tactic influenced metrics or other indicators for that attribute. When all sources agreed on the impact's direction, either *predominantly positive* or *negative*, we reported it as such. If sources reported both predominantly positive and negative impacts for the same tactic-quality attribute combination, depending on conditions of

the environment or domain, we marked the impact as *ambivalent*. In cases where no evidence of a correlation between an AT and a QA was found, we noted this absence of evidence.

Data extraction consistency. Towards ensuring data extraction consistency, we took three measures.

First, we followed the specific advice from the Kitchenham guidelines for performing SLRs [20]. According to them, it sufficient to conduct *"a test-retest process where the researcher performs a second extraction from a random selection of primary studies"*. This second extraction was conducted by Author 1 on a random sample of *10 papers* for RQ1, *15 papers* for RQ2, and *20 papers* for RQ3. The results of this extraction round were identical to the previous attempt for all RQs.

Second, we continuously discussed the data synthesis and its results in the group of authors. Author 1 strictly followed selected search and data synthesis strategies for RQ1, RQ2, and RQ3 sequentially. Whenever a synthesis of results for a particular RQ was completed, a group discussion with all authors was organized. Author 2 and Author 3 based on their expertise provided feedback on whether the search strategy was executed correctly and whether extracted QAs, ATs, or trade-offs corresponded to selected definitions and conditions for their extraction. At each meeting, the review protocol was presented and updated based on the results of the discussion.

Third, the used literary sources are shared in the publicly available supplementary artifact allowing other researchers to follow our algorithm and analyze selected papers. This also enhances the reproducibility of this research.

3.6. Results Verification

All the results should be verified by the experts and practitioners to check their relevance for industrial use. We followed several scenarios of validation depending on the contribution.

Our findings for RQ1 which were compiled in the format of the quality model were verified through:

1. *Expert Validation.* The model was presented at the Swedish Requirement Engineering meeting (SiREN 2023). This event brought together academic and practical experts with a background in the field of requirements engineering and machine learning. An assessment was or-

ganized in a focus-group setting with oral feedback. Six experts were surveyed sequentially on three main questions:

- If the proposed model is 'emphcomplete, i.e. the identified quality attributes exhaustively characterize the quality of ML-enabled systems.
- If the proposed model is *general*, i.e. the identified quality attributes are applicable to all types of ML-enabled systems, not only to a certain one.
- If the proposed model is *relevant*, i.e. the identified quality attributes respond to current challenges in ML-enabled software quality assurance.

2. *Practitioner Validation.* The model was presented to four ML engineers from Swedish AI software companies. They checked the proposed model against the key quality characteristics used in their enterprise when designing AI-based systems. The validation used the same evaluation parameters as in the case of expert assessment: completeness, generalizability, and relevance.

The findings for RQ2 which were combined in the final list of architectural tactics and associated quality attributes were verified through practitioner validation. The list of ATs was presented to four ML engineers from Swedish AI software companies. They assessed the applicability of architectural tactics to solve problems encountered in the design of AI-based systems within their company, as well as their theoretical validity for improving system qualities.

The findings for RQ3 which were summarized in the resulting table of trade-offs were verified through internal peer-reviewing, where each co-author checked the plausibility of the identified impact (or absence of such) based on their expertise. This review step did not result in any changes to the findings. An additional verification by practitioners and experts is desirable, however, it is overly laborious for the current study due to the large number of impacts identified. In Section 5, we propose and discuss a strategy for such validation in future work.

4. Results

4.1. RQ1: Identification of Common Quality Model

We examined 37 scientific sources to obtain a comprehensive list of quality attributes that characterize various ML-enabled systems. Table 1 provides a list of all quality attributes found and the number of their occurrences in all the sources studied. The list is sorted in descending order of occurrences *(#occ.)* of the quality attribute in the papers.

Table 1: All retrieved quality attributes of ML-enabled systems

QA	**#occ.**	**QA**	**#occ.**	**QA**	**#occ.**	**QA**	**#occ.**
Fairness	19	Efficiency	12	Ethics	6	Completeness	2
Safety	19	Usability	11	Data quantity	6	Consistency	2
Security	18	Accuracy	10	Traceability	4	Compatibility	2
Explainability	18	Testability	10	Legal	3	Accountability	1
Privacy	17	Correctness	9	Reusability	3	Justifiability	1
Reliability	16	Func. suitability	8	Interoperability	3	Autonomy	1
Performace	16	Interpretability	8	Reproducibility	2	Modifiability	1
Transperancy	14	Trustworthiness	8	Integrity	2	Elasticity	1
Robustness	13	Scalability	8	Repeatability	2	Resilience	1
Data Quality	13	Adaptability	6	Retrainability	2		
Maintainability	12	Portability	6	Modularity	1		

4.1.1. Studies based on interviews and questionnaires.

Several works built models based on the results of interviews, questionnaires, and surveys with experts.

The work of Habibullah et al. [21] contains the most complete list of quality indicators among all the papers studied. The set of QAs was formed through interviews with practitioners in the field of developing ML-enabled systems. The authors collected 37 quality attributes (system non-functional requirements) relevant to product operation, product revision, and product transition, such as efficiency, usability, portability, etc.

Vogelsang [26] identified the structure of common requirements for ML-enabled systems: functional and non-functional based on the interview results of several data scientists. The group of non-functional requirements (= quality attributes) included: explainability, freedom from discrimination (= fairness), legality, data quantity, and data quality.

To build sustainable AI architectures Kästner et al. [27] indicated six main characteristics of quality assurance based on expert assessment: performance, data quality, testability, safety, security, and fairness.

Agca et al. [28] conducted a comprehensive survey on trusted distributed artificial intelligence. The focus of that paper was not on creating a certain quality model, however, the research addresses such quality attributes as performance, robustness, and transparency.

Various quality models have been proposed by other authors: based on an interview study with ML-project stakeholders [29], [30], industry experts [31], [32], and based on the mixture of qualitative and quantitative studies including a survey of practitioners [33].

4.1.2. Studies based on expert assessments.

There is a group of work presenting the quality characteristics of ML-enabled systems axiomatically, i.e. the authors list them as relevant or discuss their relevance based on their own expertise. Since we are examining exclusively scientific "white" literature, we consider the authors as experts and find it reasonable to include such works in the list as well.

Yap [34] stated that ML systems have unique requirements arising from the interaction with humans such as fairness, privacy, safety, and security (covering the ML component and overall system security). The key quality requirement in that context was trustworthiness.

Ozkaya [35] pointed out that all the knowledge and experience in designing and reasoning about software systems does not immediately apply to AI-system engineering. The author suggested security, usability, privacy, explainability, data quality, and quantity, testability, and robustness as the critical attributes in the successfully designed structure and behavior of AI-enabled systems.

Zhang et al. [36] provides a comprehensive survey of techniques for testing machine learning systems. Authors defined quality attributes as testing properties, which included correctness, memory and energy efficiency, robustness, and others.

Truong [37] suggested applying the author's R3E approach to evaluate the state of end-to-end ML systems. The R3E approach consists of robustness, reliability, resilience, and elasticity.

Kuwajima et al. [38], [15] state that all quality attributes from the standard SQuaRE model could and should be applied to the development of ML-enabled software with additional quality attributes from ethics guidelines for trustworthy AI from the European Commission.

Horkoff [39] summarized a selection of quality attributes presented previously in the work of Habibullah et al. [21] and created another quality model

consisted of eight general quality attributes: accuracy, performance, fairness, transparency, security, privacy, testability, and reliability.

Various quality models have been proposed by other authors: particularly for AI-chatbots [40], ML-based systems for Automotive OEM [41], Deep Learning Systems [42], IoT systems [43], Regression-Based ML-systems [44], and other types of AI-based systems [45], [46], [47], [48], [49], [50], [51].

4.1.3. Studies based on the other methodologies.

Some papers stem from methodologies, for example, experience reports from particular companies, design science research of particular solutions, applications of common standards to specific cases, or studies of community trends.

Based on the priorities of a particular company in ML-enabled systems development, Cysneiros et al. [52] identified several key quality attributes: trust (a.k.a. trustworthiness), ethics, and transparency.

Washizaki et al. [53] collected "good/bad" software engineering design patterns for ML techniques to provide developers with a comprehensive classification of such patterns. Their patterns are implemented to directly affect quality attributes, such as performance, reliability, accuracy, and others.

Ahmad et al. [54] noticed that industry practices use tools that do not enforce requirements engineering for AI and that there are gaps between research and practices in RE for AI. They conclude that the engineering of AI-systems introduced new specs that did not exist in traditional software, which include data quality, data quantity, accuracy, and explainability.

Felderer et al. [55], [56] brought together best practices written by software engineers and data scientists. Key quality attributes according to the studies above were: data quality, system accuracy, correctness, interpretability, etc.

Arseniev et al. [57] applied fundamental software engineering principles to AI systems. They analyzed how various software teams build software applications with customer-focused AI features and which main problems they meet. The authors claimed that a substantial amount of effort is usually spent on data collection and data preparation. Data quality characteristics also reflect the quality of the AI system. In addition to data quality and quantity, the authors worked with the reliability, scalability, and convenience of accompaniment (in the context of the research equals maintainability).

Other practical-oriented solutions described in the scientific literature

were a bug benchmark [58] with key affected attributes of testability, traceability and functional suitability, quality assessment and criteria analysis for AI image recognition software [59] with an emphasis on data quality and robustness, compositional approach to creating architectural frameworks for distributed AI systems [60], explaining models in AI [61], ontology-based modeling and analysis of trustworthiness [62], ensuring dataset quality [63] and other works that operated with quality attributes [64], [65].

4.1.4. Synthesized quality model.

A contextual cut-off line between "frequently mentioned" and "infrequently mentioned" attributes was drawn based on the number of their mentions in the scientific literature according to the rule: "If the number of occurrences was greater than 4 then the quality attribute was recognized as frequently mentioned, otherwise, as infrequently mentioned".

The next step was to combine semantically similar frequently mentioned attributes into common quality attributes (CQAs).

Some authors used the term "performance", which implied "system accuracy" or "resource efficiency" depending on the context and to avoid misunderstandings, we divided this term into the above two groups during the data extraction process.

We noticed that the papers that mentioned the quality attribute of "efficiency" used it to describe four different cases: efficiency in terms of running time, efficiency in terms of memory costs, efficiency in terms of energy consumption, or accuracy of system output. The first three cases were exceptionally considered as subcharacteristics of resource efficiency, while the last one was also included as the "system accuracy".

A review of the literature showed that "correctness" and "functional suitability" were often used as contextual synonyms; "usability" was a separate attribute from any other; terms "trustworthiness", "reliability", "safety", "robustness" and "scalability" were of the same nature; "privacy" was presented as a subset of "security", "maintainability" consisted of "system transparency", "testability" and "maintainability" itself; "portability" and "adaptability" were the attributes of the same nature; "explainability" and "interpretability" described by highly-related spectrum of issues, "fairness" and "ethics" were used as synonyms with the rare exceptions of non-standard terminology when "fairness" characterized the "trustworthiness" of model's predictions; "data quantity" was often considered as a special characteristic of the overall "data quality".

The result of the semantic unification of frequently mentioned attributes into common quality attributes and their sub-characteristics is presented in Figure 1.

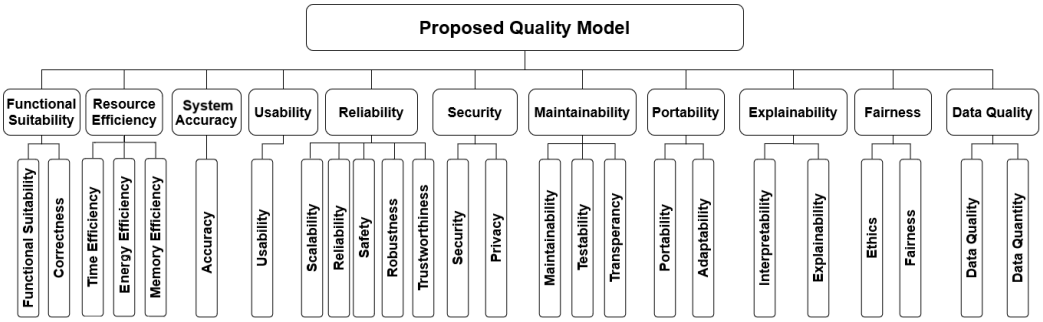

Figure 1: Proposed Quality Model for ML-enabled Systems

Our quality model comprises the following high-level common quality attributes: *Functional suitability* is the degree to which a system corresponds to functional requirements. *Resource efficiency* is the degree to which a system fulfills a given functionality within an existing amount of hardware capacities. *System accuracy* is the degree to which a system performs and analyzes the contextual environment and refers to the performance of the entire system in real-world conditions, including model inference and data post-processing. Particularly, system accuracy is different from model accuracy, which specifically measures the predictive performance of the trained machine learning model on a given dataset. *Usability* is the degree to which a system can be employed by end users to achieve specified goals. *Reliability* is the degree to which a system performs specified functions under specified conditions in the fixed domain. *Security* is the degree to which a system protects information and data. *Maintainability* is the degree to which a system can be modified and supported by developers and maintainers to achieve specified goals. *Portability* is the degree of effectiveness with which a system can be transferred from one domain, software, or hardware basis to another. *Explainability* is the degree to which the behavior of a system (primarily, the behavior of ML models) and its output can be explained by humans. *Fairness* is the degree to which a system can detect and prevent an algorithmic bias created by a model. *Data quality* is the degree of integrity and sufficiency of data for model training, testing, and validation, including the reliability of the related data sources.

4.1.5. Verification of the quality model.

The developed quality model was presented to six experts at the Swedish Requirements Engineering Meeting (SiREN 2023) in Gothenburg, Sweden organized by Chalmers University of Technology. Four experts out of six participants rated the model as fully complete, general, and relevant. One participant pointed out that the proposed model lacks the "compatibility" attribute (with reference to ISO 25010:2011 [3]). According to the used methodology, this attribute was rarely reported in the literature (only 2 times). This fact did not allow us to include it in the final model. This fact was considered as a threat to validity in Section 5. The last expert noted that the selection of terms for quality characteristics is not as important as specific metrics and ways to achieve them, however, they fully supported the proposed model in terms of completeness, generalizability, and relevance.

Further, the model was presented to four ML engineers from Swedish AI software companies. They conducted a theoretical comparison of the quality attributes we found with the system qualities considered by them when designing real AI solutions. Three of the practitioners concluded that the resulting model fully exhausts the quality of all the systems developed in their company and is relevant nowadays. The last expert noted the importance of compliance of the developed solutions with all the quality attributes we identified, as well as with the business goals of stakeholders. According to our findings, meeting business requirements can be expressed in both functional and non-functional requirements. Each NFR can be ranked according to the identified quality attributes, while strict adherence to the functional requirements corresponds to the identified "functional suitability" attribute. After clarifying the context and terminology, the expert agreed that the proposed model was complete, general, and relevant.

4.2. RQ2: Architectural Tactics

In Table 2 we present our findings under RQ2 and provide a correspondence in the format of "quality attribute - architectural tactic(s)". We highlight that the table presents only common quality attributes without subcharacteristics, but they were considered in the search strategy, the details of which can be found in the supplementary artifact[1]. Each AT is characterized by its scope as either training system (TS), deployed system (DS)

[1]Supplementary Artifact: https://figshare.com/s/57b4fa3f53caecd4a5b1

or both. Note that the TS can generally be included as a subsystem in the DS (e.g., in systems that support retraining); in such cases, DS refers to the tactic being applicable to *other* subsystems than the TS.

We note that the "functional suitability" attribute was not included in the summary table for two reasons: this attribute is general in meaning (all ML-enabled systems must follow functional requirements), but not general in the ways of its achievement (for each system there is a unique set of functional requirements), and also because we were unable to find common architectural tactics to satisfy this attribute with the selected methodology.

Table 2: Architectural Tactics to Achieve Common Quality Attributes (CQA)

CQA	Tactic	Scope	CQA	Tactic	Scope
Resource Efficiency	- Distributed Learning	TS	**Data Quality**	- Data Preprocessing	TS/DS
	- Federated Learning	TS		- Data Profiling	TS
	- Automated Data Reduction	TS/DS			
System Accuracy	- Automated Hyperparameter Tuning	TS	**Explainability**	- Local Interpretable Models	TS/DS
	- Automated Algorithm Selection	DS		- Rule-based Models	TS/DS
Usability	- Human-in-the-Loop	TS/DS	**Maintainability**	- Componentization	TS/DS
				- Containerization	DS
Security	- Intrusion Detection	TS/DS	**Fairness**	- Automated Bias Mitigation	DS
	- Automated Data Encryption	TS/DS			
	- Federated Learning	TS			
Reliability	- Automated Data Versioning	TS/DS	**Portability**	- Containerization	DS
	- Human-in-the-Loop	TS/DS		- Componentization	TS/DS

TS = Training System; DS = Deployed System

Below we provide detailed explanations of all the tactics found. To introduce a basic understanding of how each tactic can be implemented, we supplement all the tactics with one example of its implementation from the literature.

4.2.1. Tactics associated with Resource Efficiency.

The ATs to increase resource efficiency were aimed at distributing and reducing resource-intensive processes. The first architectural tactic we found was an approach to distribute the powers for model processes among several computers as known as *Distributed Learning*. Shi et al. [66] described distributed deep learning as a time and resource-efficient approach to designing the machine learning process. Considering the context in which this tactic is useful, Rao et al. [67] based on the experiments concluded that distributed learning is effective when there is a need to allocate training loads evenly. The implementation of distributed logic itself (building extra connections) does

not sufficiently affect the overall resources consumed. Other papers mentioned that traditional centralized architectures are time-consuming in the domains of biomedical images [68], photonic nanostructures [69] processing and could be replaced with the distribution of loads.

For example, Rao et al. [67] developed a distributed learning mechanism that enables self-adaptive resource provisioning by treating each virtual machine as a highly autonomous agent that submits resource requests based on its benefit.

The next explored tactic was *Federated Learning (FL)*. In contrast with the centralized approach, FL shifts the computational load to the user equipment. While distributed learning involves training models collaboratively across decentralized nodes with mostly shared data, federated learning specifically focuses on training models mostly on local data. Considering application contexts, FL can be profitable when there is a need to lighten the load on the server [70] during model training.

For example, Brisimi et al. [71] developed a federated learning framework that can learn predictive models through peer-to-peer collaboration without raw data exchanges solving a binary supervised classification problem to predict hospitalizations based on clients' data.

Finally, we found the tactic of *automated data reduction* relevant for resource efficiency. This tactic can be used in the ML pipeline to reduce training data [72] or to reduce large amounts of input data in real-time when the system is deployed and operates [73]. Considering application context, this tactic is useful in systems with massive amounts of data, such as those in the Internet of Things (IoT) [72] domain. The main perspectives of big data reduction are noise-cleaning and addressing the "curse of dimensionality" caused by millions of variables in big datasets [73]. In terms of limited resources, this tactic can be the only way to run the system [74].

For example, Rehman et al. [73] collected different methods of automated data reduction in the form of big data compression algorithms, dimension reduction methods, and redundancy elimination.

4.2.2. Tactic associated with Usability.

With our search strategy we could identify only one AT for increasing the usability of ML-enabled systems, which was the integration of a human as a system component, so-called *"human-in-the-loop" (HitL)*. Petrelli et al. [75] stated that usability indicators of AI-based systems were sufficiently improved after the integration of so-called "interaction designers"

into the processes. They acted as experts to coordinate reciprocal understanding. Winter et al. [76] and Kröll et al. [77] viewed close interaction and co-integration between users and the model as mutually beneficial. The user becomes more proficient in using machine learning for their tasks and gets a clearer picture of basic AI principles, while their feedback can serve as a basis for improving the system in terms of usability as well as fairness, explainability, and data quality. According to Heimerl et al., [78] integration of experts in the training system (when the experts manually evaluate and update training datasets) is as beneficial in terms of usability as their integration in the deployed system (when they evaluate system outputs). Sperrle et al. [79] proposed a human-centered approach to the evaluation of ML-enabled systems and highlighted the necessity of HitL to improve system usability. Considering the application context, the tactic is especially relevant for accessibility-critical systems [75] or systems primarily oriented toward the end users [78].

For example, Gomez et al. [80] noticed that the ML-enabled system in a specific case overlooked human factors (such as human workload or timing) which were critical for end-users. To address this issue, the model was replaced with an adaptive one to consider parameters provided by specific users.

4.2.3. Tactics associated with Reliability.

To address system reliability or its sub-charateristics we found an AT of *Data Versioning*. Lewis et al. [81] conducted a complex study on architectural challenges in ML-enabled systems and among other fundamental conclusions, stated that the technique of data versioning can be effective in improving reliability and robustness that serves as a safety net in case of unexpected failures during data processing. This aspect covers mostly the training system. However, in addition to versioning of datasets, it is also important for architects to consciously version related artifacts, such as parameters for model training, data for model evaluation, and evaluation results ("co-versioning") [82]. Warnett et al. [83] proposed the versioning of output data when the system is operating to ensure the traceability and integrity of results, allowing for the accurate tracking of changes in outputs over time. Considering the application context, such a tactic is relevant for all systems where multiple versions of a model could be deployed, either over time or even in parallel [84].

For example, Van et al. [82] used data versioning in the format of sav-

ing and storing different versions of end-to-end machine learning pipelines (including datasets for data processing pipelines and model coefficients for model training pipelines) to ensure that multiple versions of a pipeline can run in parallel.

Another tactic to address reliability and safety concerns was previously introduced *Human-in-the-Loop (HitL)*. Rajendran et al. [84] stated: "The involvement of humans during the training phase can play a crucial role in mitigating some safety issues of autonomous systems", although it can also lead to extra expenses for the vendor. Considering application contexts, integrating expert users is reasonable to validate solutions that need to be available at sufficient capacity [84].

For example, Rajendran et al. [84] explored different integrations of experts as components to improve the reliability of deep learning autonomous systems such as "learning from demonstration", "learning from intervention", and "learning from evaluation" to deal with unforeseen circumstances and define safer policies.

4.2.4. Tactics associated with Security.

The most frequently considered AT for improving security was the integration of *Intrusion Detection*. Sanju [85] claimed: "The protection of IoT systems from attacks and the assurance of their security posture is ensured by intrusion detection systems". Liu et al. [86], Qu et al. [87], Laqtib et al. [88], Rashid et al. [89], Balbali et al. [90] listed several fundamental benefits of using intrusion detection as microservices in the architecture of industrial IoT for enhancing security. Intrusion detection is mainly used for preventing data poisoning (where exclusively the training system is under attack)[89] and adversarial attacks (where minor malicious changes in input data can cause the model to make incorrect predictions) [87]. In comparison with traditional software, intrusion detection in ML-enabled systems is more flexible and adapted to changes due to possible different outputs produced by the model over time or new behavior of the model caused by retraining. Considering the application context, the tactic is useful for systems operating with large amounts of real-time input data or big datasets consumed by training system [88], including IoT systems [85].

For example, Sanju et al. [85] introduced a hybrid metaheuristics-deep learning approach with an ensemble of recurrent neural networks to detect and prevent intrusions in real-time data processing in an operating IoT system.

Another identified AT was *Automated Data Encryption.* McGraw et al. [91], Bekri et al. [92] when analyzing risks for the security of ML-enabled systems, highlighted the important role of encryption of training and testing datasets to protect from several threats primarily. Wu et al. [93] found big data encryption efficient for system security, however, some encryption methods may not be optimal also for privacy by default. Kantarcioglu et al. [94] along with several non-architectural decisions, found data encryption as one of the ways to satisfy security requirements in industrial solutions when it comes to ensuring the security of all data flows within a deployed system. In addition to the encryption of datasets and input data, the encryption of the model parameters and weights should be conducted. Considering the application context, the implementation of this tactic is especially relevant for highly sensitive systems that operate continuously [92], including privacy-preserving deep learning systems [95].

For example, Phong et al. [95] proposed additively homomorphic encryption to increase the privacy and security of neural networks by allowing computations to be performed on encrypted data without decrypting it, thus protecting sensitive information throughout the processing stages.

Finally, an effective tactic to architecturally increase privacy and security is an implementation of *Federated Learning (FL).* In addition to the benefits of this tactic for resource efficiency, it is commonly used to "train a massive amount of data privately due to its decentralized structure" [96]. Zhou et al. [97] proved the statement above: "The emerging federated learning (FL) offers a feasible solution for the privacy preservation of users' sensitive data in training AI models". In other words, federated learning allows the benefits of data privacy without the need for data to be shared with a central server [98], [99]. Considering the application context, this tactic is useful for privacy-critical systems [98, 99], including personalized big data systems [97].

For example, Zhou et al. [97] implemented a federated learning algorithm that ensures that sensitive data is not disclosed during model training together with a user-level personalized differential privacy mechanism.

4.2.5. Tactics associated with Maintainability.

For both traditional and ML-enabled systems, the architectural tactic of *Containerization* has shown its effectiveness in terms of maintainability and system transparency. Rovnyagin et al. [100] explicitly claimed the positive effect of containerization along with related tools (such as docker or orchestrator) on system maintainability and operability. Kolltveit et al. [101]

stated: "Models packaged in containers are simply run directly as standalone services" which contributes to the enhancement of maintainability. According to Openja et al. [102]: "Docker (which is a containerization service) allows for convenient deployment of websites, databases, applications' APIs, and machine learning (ML) models with a few lines of code". Finally, containerization allows applied scientists without advanced knowledge to deploy models and access High-Performance Computing (HPC) [103]. Considering the application context, this tactic is useful for systems that require more isolated dependencies and simplified updates [100, 101, 102]

For example, Openja et al. [102] identified 21 major categories representing the purposes of existing ML projects using Docker, including those specific to ML models, which in turn reduces the complexity of managing ML models.

Another efficient tactic for enhancement of system maintainability was *Componentization*, which obviously contributes to more transparent testing [104]. The componentization can be applied to the overall deployed system architecture, or exclusively to the training system, or even exclusively to a model. Singravel et al. [105] stated that "Component-Based Machine Learning (CBML) enhances the capabilities of the monolithic ML models" in terms of transparency. Considering the application context, this tactic is relevant for systems that require constant monitoring and updates from the side of developers or maintainers.

For example, Singravel et al. [105] transformed the monolithic ML model in the domain of space exploration into a set of connected components which simplified its further maintenance.

4.2.6. Tactics associated with Portability.

According to the literature review, both tactics associated with maintainability were also profitable for portability.

Componentization is a relevant architectural tactic for increasing portability [106]. Shadab et al. [107] reported that the development of AI logic in the format of reusable components could be an adequate solution to increase portability, however, it also may introduce new risks since "currently there is no framework that guides the selection of necessary information to operate in a system different than the one for which the component was originally purposed". Geyer et al. [108] concluded that components instead of one monolithic model extend reusability and generalization, which in the context of our research directly contributes to the quality of portability. In terms

of portability, both componentization of the overall architecture as well as dividing the ML model into components are profitable. Considering the application context, this tactic is useful for cross-platform compatible systems that are partly or fully transferred from one software or hardware base to another [106] as well as to systems that are transferred from one environment to another [108].

For example, Geyer et al. [108] replaced monolithic models with a component-based approach that develops machine learning models not only for the parameterized design of the whole buildings in the construction domain but also for the design of its semi-independent parts on the lower level of abstraction.

The tactic of *Containerization* also improves the portability of ML-enabled systems by encapsulating all dependencies and configurations [109]. Considering the application context, this tactic is useful for cross-platform compatible systems, including cloud-based services [110].

For example, Naydenov et al. [109] studied different container technologies used in ML-enabled systems, such as K8s, K3s, Docker, Rancher, and others allowing the systems to run consistently across different platforms, such as local machines or cloud servers.

4.2.7. Tactics associated with Explainability.

Local Interpretable Models (LIME) are a tool for solving issues of explainability (XAI) and interpretability [111], [112], by approximating the behavior of a complex underlying model around a specific prediction using a simpler local model that can *explain* the prediction. The intended application context of LIME is a system with a "black-box" model that is inherently non-interpretable, such as a neural network obtained by deep learning [113]. Such complex models cannot be avoided in domains that deal with particularly complex phenomena, such as weather forecasting [114] and intrusion detection [115], where LIME is particularly useful. LIME is model-agnostic, which means it can be applied to any ML model without knowing its internals as it only requires access to prediction probabilities. Integrating LIME into a training system can help explain how the model comes to conclusions before deployment and help with the selection of the final model to be used in production. It can be also used in an already deployed system to explain the behavior of the existing model [116].

For example, Saadatfar et al. [116] proposed a LIME algorithm that generates more focused data samples close to the decision boundary and simultaneously close to the original data point in comparison with different LIME

implementations, such as BayLIME, SLIME, and LS-LIME.

Another widespread tactic found was the usage of *Rule-based Models* [117], [118]. While LIME explains individual predictions of complex models by fitting simpler models locally around specific instances, rule-based models directly encode prediction rules that are clear for a human [119]. The intended application context is one where that permits such rules to be formulated, which then inherently leads to explainability and interpretability, and can make rule-based models favorable over more complex ones, especially for non-complicated tasks [120]. Rule-based models can also be used for rule-based approximation and visualization [121]. This AT can be also used in both training and deployed systems [122].

For example, Rajapaksha et al. [122] developed a model-agnostic rule-based approach that obtains k-optimal association rules from a neighborhood of the instance to be explained.

4.2.8. Tactics associated with System Accuracy.

Automated Hyperparameter Tuning can be especially profitable in increasing model(s) accuracy resulting in the enhancement of the overall system accuracy. It is well-known that the accuracy of machine learning models relies on hyperparameter tuning [123]. Daviran et al. [124] stated: "The predictive accuracy of models can significantly increase when the optimized hyperparameters are predefined and then adjusted to training procedure", which, in this tactic, is an automated process. Considering the application context, this tactic is especially valuable for systems with complex models or large datasets where manual tuning is ineffective, including deep learning systems [125].

For example, Ottoni et al. [125] proposed a framework for automated hyperparameter tuning and based on the experimental results proved that this tactic sufficiently improved different accuracy metrics of an image recognition deep learning system.

Another tactic for system accuracy is an *Automated Algorithm Selection.* Kerschke et al. [126] proposed their implementation of automated algorithm selection from a pre-defined set of algorithms and noted that the choice might be made not only to maximize the accuracy but also based on other contextual priorities. Pise et al. [127] listed several key factors that must be considered in a proper algorithm selection tactic. Such a tactic fundamentally improves accuracy in healthcare and medical systems as well [128], [129]. Considering the application context, this tactic is generally useful for the

systems in which high accuracy is particularly important [126, 127, 130], and more specifically in systems operating in constantly changing environments [130].

For example, Alissa et al. [130] proposed a technique for automated algorithm selection, applicable to certain optimization domains in which implicit sequential information is encapsulated in the data. Specifically, they trained two types of recurrent neural networks to predict a packing heuristic.

4.2.9. Tactic associated with Fairness.

Automated Bias Mitigation is a common term for a set of algorithms developed to increase the fairness of the deployed ML-enabled system outputs. Lee et al. [131] conducted a review of so-called fairness toolkits with the analysis of their relevance in improving system outputs from the ethical perspective. Ferrara et al. [132] suggested that "building specific methods and development environments, other than automated validation tools, might help developers treat fairness throughout the software lifecycle". Other algorithms for automated bias detection and mitigation were proposed by Agarwal et al. [133], Castelnovo et al. [134] and Zhang et al. [135]. Considering the application context, this tactic is primarily useful for the systems operating with personal data and sensitive parameters [113], in particular those that are not inherently interpretable, such as deep learning systems [113].

For example, Maan et al. [113] proposed a method that evaluates the fairness of deep learning model behavior against sensitive attributes (i.e. age, race, gender, sex, and so on) to help mitigate biases without compromising much on accuracy.

4.2.10. Tactics associated with Data Quality.

The tactic of *Automated Data Profiling* is considered an effective tool to increase data quality in the domain of ML-enabled systems [136]. Data profiling contributes to the training system allowing the identification of missing values and the detection of outliers and anomalies. Considering the application context, this tactic is generally useful for systems that deal with heterogenous and complex data and, therefore, require comprehensive evaluation to ensure sufficient data quality for training a high-quality model, including domains such as cybersecurity [137], digital twins [138], and healthcare systems [139].

For example, Pansara et al. [140] proposed to employ extra machine learning algorithms to automatically profile and cleanse master data for complex

model training operating in the domain of environmental sustainability.

While data profiling implies examining the structure and the content of data to understand its features, *Automated Data Preprocessing* includes data cleaning and data transforming to prepare it for analysis. Gawhade et al. [141] and Ramkumar et al. [142] proposed computerized data preprocessing algorithms to primary process input data before it enters the ML model in the deployed system. Santos et al. [143] suggested another implementation of automated data preprocessing used for the preparation of training datasets in supervised machine learning. In terms of ML-enabled systems, data preprocessing includes data splitting (dividing data into training, testing, and validation datasets). Considering the application context, this tactic is necessary for all types of ML-enabled systems that are going to be trained on unprepared datasets [143] or that operate with raw data on the input [142].

For example, Bilal et al. [144] proposed an automated pipeline for advanced data preprocessing steps of target discretization and sampling which are validated using RandomForest.

4.2.11. Definitions of Architectural Tactics.

Below we present brief contextual descriptions of all architectural tactics found. These definitions were built based on the experience from literature and brought to the common format (relevant for all studied papers despite specifics and context).

AT1: Distributed Learning is an architectural approach to machine learning aimed at parallelizing computing powers among several computers [67].

AT2: Automated Data Reduction is an automated process aimed at minimizing the complexity and size of datasets while preserving their essential information (widespread in IoT) [72].

AT3: Federated Learning is an architectural approach to machine learning aimed at training on local heterogeneous datasets [70].

AT4: Human-in-the-Loop (HitL) is an architectural approach where a human (expert) is integrated into the ML-enabled system as a separate component aimed at monitoring and improving the system's behavior [75].

AT5: Automated Data Versioning is an automated process aimed at the creation, tracking, and management of different versions or iterations of datasets used for model training, testing, and validation [83].

AT6: Intrusion Detection is a tactic for complex systems (primarily, IoT systems) aimed at the detection and classification of network intrusions and

anomalies [85].

AT7: Automated Data Encryption is an automated process aimed at securing sensitive data used for training, inference, or model deployment to protect it from unauthorized access [91].

AT8: Containerization is an architectural approach aimed at packaging an entire system or model (incl. its dependencies and runtime environment), into a standardized unit called a container [100].

AT9: Componentization is an architectural approach aimed at breaking down a software system into modular components or building blocks that can be independently developed, tested, and deployed [106].

AT10: Local Interpretable Models (LIME) is an approach to machine learning aimed at explaining black boxes by approximating the behavior of a complex model around a specific prediction using simpler (more interpretable) models [111].

AT11: Rule-based Models is a type of model that relies on explicit rules (i.e. if-then) that are designed and specified by humans or domain knowledge to approximate complex model behavior [118].

AT12: Automated Hyperparameter Tuning (or Hyperparameter Optimization) is a method aimed at searching for the best hyperparameter values for the model based on certain criteria [125].

AT13: Automated Algorithm Selection (or Algorithm Configuration) is an automated process aimed at searching the most appropriate method(s) for a certain task or in certain circumstances [126].

AT14: Automated Bias Mitigation is an automated process aimed at identifying and reducing bias in algorithms, models, and datasets by their evaluation through fairness metrics or "sensitive" feature monitoring [132].

AT15: Automated Data Preprocessing is an automated process aimed at preparing raw data for analysis and model training [136].

AT16: Automated Data Profiling is an automated process aimed at analyzing and summarizing the characteristics of a dataset to gain insights into its structure, quality, and distribution [141].

4.2.12. Verification of the architectural tactics.

All authors of this article were conducting constant peer-reviewing of the resulting list of ATs. During weekly meetings, architectural tactics were discussed against identified quality attributes based on the expertise of each co-author. During the validation, issues related to the architectural nature

of the identified artifacts and the advisability of classifying them as tactics were discussed. In other words, based on the studied literature we checked if the artifacts affected the overall principle of architectural design (e.g., componentization) or could be integrated as constituent parts into the overall system architecture (e.g., automated bias mitigation module). In this paper, we presented a list of ATs agreed upon by all co-authors of this work.

Further, the list of ATs was shared with four ML engineers from Swedish AI software companies. One practitioner stated that he had experience with all of the suggested tactics to "a greater or lesser extent" except federated learning. He concluded that, based on his experience, all of the tactics presented were relevant to the quality attributes associated with them with an exception for federated learning. Due to insufficient expertise, the interviewee could not confirm or deny this connection. The second practitioner had a similar background and experience with all of the tactics listed except federated learning. He concluded that the list of tactics was accurate and consistent with quality attributes, however, he noted that the list was not complete. He proposed supplementing the list with the tactic of "code versioning" to improve reliability and maintainability. Using our methodology, we were unable to find this tactic relevant in the literature. However, from a practical point of view, we see the importance of this remark. It requires additional assessment and refinement of the search string. The other two experts confirmed their experience in employing all the listed ATs and found all of them relevant in the context of corresponding QAs. They provided several organizational decisions on how to improve resource efficiency and fairness (e.g., evolution of developing culture), however, they struggled to propose any additional ATs to this list.

To enhance the generalizability of verification results, it is preferable to continue validating the list of ATs by experts and practitioners. We anticipate that the list of architectural tactics will expand as we receive feedback from experts. We see great potential in updating the list of tactics and further exploring new entries.

4.3. RQ3: Trade-off analysis

Table 3 represents the summary of our findings obtained during the systematic literature review to answer RQ3.

Below we provide an analysis of the papers which investigate quality trade-offs of the identified architectural tactics.

Table 3: Trade-off analysis: impact of Architectural Tactics on Quality Attributes

Quality Attribute	AT1	AT2	AT3	AT4	AT5	AT6	AT7	AT8	AT9	AT10	AT11	AT12	AT13	AT14	AT15	AT16
Resource Efficiency	+	+	+	0	0	-	-	-	0	+	0	-	-	-	a	0
Usability	+	0	0	+	0	0	0	0	0	0	0	0	0	0	+	+
Reliability	0	0	a	+	+	+	+	+	+	-	0	+	+	a	-	+
Security	a	0	a	a	0	+	+	-	0	0	0	+	0	0	+	+
Maintainability	0	0	0	+	+	0	0	+	+	+	0	0	+	0	+	a
Portability	+	0	0	0	0	0	0	+	+	0	0	+	0	+	0	0
Explainability	-	0	0	+	0	-	0	0	+	+	+	0	-	0	+	0
System Accuracy	0	-	-	0	0	0	-	0	+	-	-	+	+	-	a	+
Fairness	-	0	-	+	0	0	0	0	0	+	0	+	0	+	a	0
Data Quality	0	-	-	+	0	+	+	+	0	0	0	0	0	a	+	+

+ = predominantly positive impact, − = predominantly negative impact, a = ambivalent impact, 0 = insufficient evidence either way

4.3.1. AT1: Distributed Learning.

Distributed learning can have positive side-effects on system usability by enabling learning across multiple nodes, thereby enhancing responsiveness and adaptability to diverse user needs [145]. However, the impact of distributed learning on privacy (in the current context: on security as well) is controversial. On the one hand, due to their distributed nature such systems are more stable in terms of security since they do not rely only on one server [146] and they can be profitable to preserve privacy due to the essence of decentralized nodes without a necessity to share sensitive information centrally [147]. On the other hand, issues with data confidentiality, security breaches, and potential misuse of personal information are connected to increased exposure of sensitive data across those decentralized nodes [148], [149]. The positive impact of distributed learning on portability is proved by its ability to transfer and deploy trained models across different computing environments and devices [145]. The negative impact of distributed training on explainability arises from the difficulty of tracking and understanding how individual augmented data from different nodes influence the final results of the model [150]. Finally, representation across decentralized nodes can lead to biased model outputs and unequal treatment of different demographic classes [151].

4.3.2. AT2: Automated Data Reduction.

In addition to the obvious improvement in reducing the load on system resources, automated data reduction tactics also have some limitations. Any interventions in datasets can be risky, especially for complex (low-explainable) models. First of all, such a tactic may decrease system accuracy due to the potential loss of important data during the reduction process [152]. The risks

of incorrect perception of data by the algorithm and classification of useful data as noise or outlier is an obvious risk for data quality and quantity when implementing this tactic [153], [154].

4.3.3. AT3: Federated Learning.

In RQ2 we identified that federated learning is often used for increasing security and privacy particularly, however, some papers found for RQ3 by Shen et al. [155], Jeong et al. [156], Jagarlamudi et al. [157] and Shin et al. [158] point to the practical insecurity of existing implementations of federated learning and noted severe vulnerabilities associated with data leakage or inference attacks during the decentralized model training across multiple devices: "existing federated learning protocol designs have been shown to be vulnerable to adversaries within or outside of the system, compromising data privacy" [159]. Therefore, based on the development level of federated learning at the time of writing the current paper, its impact on security and privacy is recognized as controversial. The reliability of federated learning systems can be considered ambivalent. On the one hand, "federated learning resulted in a reliable strategy for model development" [160] due to its capability to incorporate diverse and decentralized data sources. On the other hand, potential communication bottlenecks and data heterogeneity across devices lead to severe challenges in terms of robustness [159], [161], [162]. Some risks of federated learning are also connected to system accuracy due to the aggregation of diverse and potentially noisy local data from distributed devices and fairness due to insufficient diversity of data collected [163]. Such challenges also affect overall system data quality.

4.3.4. AT4: Human-in-the-Loop (HitL).

The effect of HitL on security and privacy is controversial. On the one hand, integration of human intelligence as a system component can bring the benefit in guiding the XAI-enabled system and generate refined solutions in terms of vulnerability detection [164], [165], and on the other hand, "the involvement of humans results in an external and unpredictable element that increases security concerns" [166]. Human-in-the-Loop is a unique element that plays a crucial role in the human-centered system qualities such as maintainability by intelligently tracking changes and intermediate results over time [167], explainability by leveraging bidirectional symbiotic sensing feedback [168], [169], [170] and fairness by identifying sensitive data and parameters [171], [172], [173]. Finally, data quality can be significantly im-

proved based on the feedback constantly provided by analysts and engineers [174].

4.3.5. AT5: Automated Data Versioning.

Automated data versioning can enhance the maintainability of ML-enabled systems by ensuring reproducibility and traceability of model training and inference, which simplify debugging and model updates [175], [176].

4.3.6. AT6: Intrusion Detection.

Intrusion detection in IoT systems can negatively affect resource efficiency due to the high computational and memory requirements of deep neural networks [177], [178]. While "an intrusion detection system is a promising automotive security enhancement", it also improves anomaly detection capabilities, thereby improving overall system robustness by reducing the risk of non-security-related failures and errors [179]. The inherent complexity of deep neural networks for intrusion detection negatively affects the explainability of the overall often low-explainable IoT systems [180], [181]. Finally, identifying and removing unnecessary data from the datasets significantly contribute to the data quality on a system level [90], [109].

4.3.7. AT7: Automated Data Encryption.

Data encryption in ML-enabled systems can negatively impact resource efficiency by increasing computational overhead and latency due to the additional processing requirements for encryption and decryption [182], [183]. The positive side-effect of data encryption in terms of reliability appears due to ensuring data integrity and reducing the risk of data corruption [184]. Wang et al. [185] noted a slight decrease in the system accuracy of the encrypted model in comparison with non-encrypted solution. Any data protection tactic also makes a significant contribution to overall data quality [186].

4.3.8. AT8: Containerization.

The main challenge of containerization in terms of resource efficiency arises from the potential resource overhead of container orchestration and virtualization [187], [102]. However, such a tactic has a positive side effect on system reliability by providing a consistent and isolated runtime environment [187]. Figueroa et al. [188] claimed: "Combined with IoT, containerization allows efficient allocation, fast execution, and deployment of hardware

resources". According to Joraviya et al. [189]: "Containerization has introduced new security challenges including cloud data breaches in ML-enabled systems", which is also accompanied by increased attack surface and potential misconfigurations including increasing risks of data breaches, model theft, and adversarial attacks due to shared resources, image vulnerabilities, and insufficient isolation, making strict access control and monitoring essential. Finally, it has a positive effect on data quality due to its ability to facilitate consistent data handling and processing environments [190].

4.3.9. AT9: Componentization.

With our search strategy we could not find any evidence that componentization has any crucial impact on resource efficiency. However, we found a positive impact of this tactic on the reliability of IoT systems [191] by enabling the implementation of safety-critical components with clear interfaces and well-defined behaviors. At the same time, this tactic is reasonable to isolate specific model components to improve explainability and interpretability [192]. Finally, based on the experimental results Heisele et al. [193] concluded that "the component system clearly outperformed global systems on all tests in terms of accuracy".

4.3.10. AT10: Local Interpretable Models (LIME).

The work of Kumarakulasinghe et al. [194] significantly contributed to the analysis of trade-offs when applying local interpretable models. According to this study, LIME provides interpretable and simple local explanations without a need for resource-intensive global model explanations, which in some cases is really profitable for resource efficiency. This tactic of simplification also contributes to improvements in maintainability. However, it goes in contrast with reliability, where LIME brings the risks of potentially incorrect local explanations that do not accurately reflect the overall behavior. Mori et al. [195] also found this fact a reason for misleading interpretations and decisions (which is considered a negative impact on system accuracy). At the same time, LIME can be used to assess a classifier's fairness and to determine the sensitive features to remove [196].

4.3.11. AT11: Rule-based Models.

The only impact of rule-based models found with our search strategy was on system accuracy, which can suffer from limited adaptability to complex

and dynamic data patterns when scenarios lie outside the predefined rules [197], [198], [199].

4.3.12. AT12: Automated Hyperparameter Tuning.

When automated hyperparameter tuning (HPT) is aimed at increasing system accuracy, the trade-off with performance efficiency occurs most often [200], [201]. Liu et al. [202] claimed: "The current resource provisioning approaches for HPT are unable to adjust resources adaptively according to the upward trends of HPT accuracy at runtime. On the other hand, dynamic resource provisioning approaches based on checkpointing are inefficient for HPT, because of the high overhead of context switching and job restarting". HPT can enhance reliability by optimizing model generalization, reducing the risk of overfitting [203], [204]. The positive impact of HPT is also noted in terms of security [205], [206] by improving resistance against adversarial attacks. Feroz et al. [207] claimed that HPT can improve not only system accuracy and system reliability but also the adaptability and portability of the system in different real-life scenarios. Finally, HPT can be used in the form of optimizing model parameters to reduce bias and consider different demographic groups or sensitive attributes [208], [209].

4.3.13. AT13: Automated Algorithm Selection.

Automated algorithm selection like HPT often brings a trade-off between resource efficiency and system accuracy [210], [211]. The main possible benefit of the automated algorithm selection component lies in the recommendation of a promising learning algorithm based on meta features computed from a given dataset [212]. Such analysis can be too complicated and time-consuming for human data scientists and delegation of those responsibilities to ML is considered a valuable contribution to system maintainability. Also, it in some sense minimizes human factors in algorithm selection, which has a positive effect on reliability as well. However, existing automated algorithm selection methods for increasing accuracy rarely consider explainability as a factor for selection, which leads to the complexity of automatically chosen algorithms [213].

4.3.14. AT14: Automated Bias Mitigation.

Hutiri et al. [214] found the risks of computational overhead due to the complexity of bias detection and correction algorithms in the context of IoT systems. The impact of this tactic on reliability is ambivalent. On the one

hand, such methods improve system stability when working with diverse datasets, however, they can also lead to potential unintended model changes with risks of system failures [215], [216]. Increased adaptability of the ML-enabled system also influences the common attribute of portability [217]. The potential distortion or removal of relevant patterns in the data introduced by this tactic harms system accuracy [214], [218]. This fact also affects the attribute of data quality [219].

4.3.15. AT15: Automated Data Preprocessing.

This tactic has a controversial impact on resource efficiency. According to Ramirez et al. [220]: on the one hand, the introduction of automated data preprocessing contributes to a faster and more precise learning process which can potentially save resources, on the other hand, when it comes to big data systems such tactic can lead to resource overload due to the large volumes of data being processed. Rendleman et al. [221] proposed a method to increase the usability of a certain module when data preprocessing is conducted according to the priorities of end users. Automated preprocessing offers certain benefits in terms of model training, such as lowering the manual effort required for data preparation and enhancing maintainability by structuring and formatting data [222], while it can also protect models against malicious inputs and data poisoning [223], [224]. It also improves explainability by ensuring consistent and standardized data transformation, which makes model behavior and decision insights clearer to humans [225], [226]. The impact of this tactic on system accuracy is ambivalent since it can be either improved by standardizing input data and noise cleaning or harmed by potentially introduced biases or distortions by the algorithms [227], [228]. The complexity of automated data preprocessing in the context of human-centered learning can also have different implications for fairness due to the same reasons [227].

4.3.16. AT16: Automated Data Profiling.

With our search strategy we were unable to find any evidence that automated data profiling has any crucial impact on resource efficiency as we expected. However, the personalization of certain data (which is a subset of data profiling) can significantly increase usability according to the needs of certain users [229], [230]. Data quality improvements provided by automated data profiling modules play a crucial role in overall system reliability [231]. In the context of IoT, data profiling can detect data vulnerabilities and privacy risks as an improvement of system security and upgrade feature

understanding as an improvement of system accuracy [232]. Finally, the impact of data profiling on maintainability is controversial since it can reduce manual effort on data management, but brings the risk of over-reliance on automated processes, which can be "black boxes" if they are executed by complex ML-algorithms [233], [234].

4.3.17. Verification of the trade-off matrix.

The resulting table of quality trade-offs was constantly being peer-reviewed by co-authors of this paper based on their independent expertise. This paper presents a version of the table agreed upon by all authors of the article.

All identified trade-offs are supported with literature references, however, more expert validation is desirable. Due to the large number of identified impacts, it would be complex and lead to significant effort. We present a strategy for such assessment in Section 5.

5. Discussion

5.1. Observations regarding quality attributes.

This study proposed a common quality model for ML-enabled systems. This model took a broader look at ML-enabled specific nature and suggested considering attributes related to "data quality" and ML-unique "explainability", "system accuracy" and "fairness" along with a standard set of attributes.

The attribute of system accuracy is a complex indicator that goes beyond model accuracy itself. In the context of quality, it is used to understand whether the system can operate effectively in the existing context with the existing accuracy (incl. metrics of precision, recall, F-score, etc.).

Thirteen papers referred to data quality as an attribute of the overall system quality. This attribute characterizes the quality of data sets used for model training and testing, as well as the ways this data was obtained and the sources from which this data was collected from at the system level. Working with unreliable or incomplete data causes a high risk of system failure due to its incorrect or insufficient perception of contextual reality as well as legal issues.

Quality attributes rarely addressed in the reviewed literature may still deserve further study. For any attribute named at least one time in RQ1, we can assume that it is relevant for some specific ML-enabled system(s). For

example, the retrainability attribute can be very important for systems operating in especially dynamically updated environments, and the autonomy attribute could be relevant for systems that, for a number of reasons, need to be isolated from all external influences.

Remarkably, the standard quality attribute of compatibility is emphasized in the literature as relevant to the quality of ML-enabled systems only two times. For this reason, this quality attribute was not included in the proposed model. According to studied papers, we noticed that the considered works typically sought to study characteristics connected to external entities: resource efficiency as a result of interaction with available resources, usability as the result of interaction with end-users, maintainability - with developers and maintainers, system accuracy - with context, fairness - with society, portability - with new environments, etc. However, the interaction of ML-enabled systems with other software systems (which is the basis for compatibility) remains poorly described in the scientific literature and likely requires additional attention.

5.2. Comparison to ISO standards.

In 2023, the International Organization for Standardization (ISO) issued a new standard ISO 25059:2023 [5]. This standard offers a quality model for AI systems, which can be seen as a possible alternative answer to RQ1. A comparison of our quality model with ISO 25059:2023, as well as with the traditional SQuaRE quality model (ISO 25010:2011 [3]), is presented in Table 4. The table maps all high-level attributes from three quality models grouped by semantic similarity. We now compare our obtained quality model to those from two relevant ISO standards. Importantly, our goal is not to present a new, competing standard, but to reflect the results of our literature review in comparison to existing quality models, in particular, those from the standards.

Factually, neither of the previous standards considers system accuracy and data quality on the system level, whereas we found in RQ1 that the literature frequently mentions them as system-level concerns and in RQ2 identified appropriate architectural tactics to address them. Moreover, ISO 25059:2023 considers the ethical perspective (related to fairness in the terminology of our research) as a high-level quality attribute and combines transparency with explainability. In our model, we decided to separate explainability and transparency. While system transparency refers to the openness and accessibility of a system's internal processes (which is mostly important

for developers and maintainers), system explainability focuses on how clearly the system's decisions and processes can be understood and interpreted by humans (which is more important for users, operators, and experts). The release of ISO 25059:2023 confirmed the relevance of RQ1 and highlighted the need to consider ML-specifics in assessing the quality of software systems.

Table 4: Comparison of proposed quality model and ISO standards

Proposed Quality Model	**ISO 25059: AI systems**	**ISO 25010: general software**
Functional Suitability	Functional Correctness	Functional Suitability
Resource Efficiency	-	Performance Efficiency
Usability	User Controlability	Usability
Reliability	Robustness	Reliability
Security	-	Security
Maintainability	Intervenability	Maintainability
Portability	Functional Adaptability	Portability
Explainability	Transparency	-
System Accuracy	-	-
Fairness	Ethical	-
Data Quality	-	-
-	-	Compatibility

5.3. Observations regarding trade-offs.

The analysis of trade-offs proposes a comprehensive mapping of different impacts after the implementation of certain ATs reported by different researchers. The most frequently reported trade-offs appeared between system accuracy and resource efficiency, system accuracy and explainability, fairness, and resource efficiency. Also, notable positive "side-effects" include the facts that: architectural enhancement of security can often increase data quality and reliability, enhancing system accuracy positively affects reliability, and tactics to improve data quality can have a positive impact on usability and security.

We met a remarkable situation with AT3 of Federated Learning. While investigating RQ2, we found its wide application to increase resource efficiency and security. However, during the work on RQ3, we identified several significant vulnerabilities in existing implementations of federated learning, which motivated us to characterize its impact on security as controversial.

Context analysis is critical when applying architectural tactics of distributed learning in terms of security, automated data preprocessing in terms

of accuracy and fairness, and automated bias mitigation in terms of reliability and data quality.

5.4. Threats to validity.

The main threat to external validity is that the generalizability of our architectural findings can be limited as we restrict our analysis to available literature, specifically, scientific papers. While the analyzed literature stems from diverse domains and methodologies, including those that analyze practical experiences from companies, we cannot claim generalizability to all possible systems and sub-domains. A planned mitigation is to conduct a grey literature study investigating non-scientific literature such as blog entries and repository documentation. Another threat to external validity is caused by the complexity of the search process for RQ2. We searched for architectural tactics with corresponding keywords, however, it is possible that some architectural tactics were not referred to as such in the literature. To mitigate this we also included related terms (e.g., "design decisions") in the search query. However, we still cannot state that the list of found ATs is complete and to mitigate this threat, we plan to conduct some more interview studies with practitioners and experts. Finally, the mapping of quality trade-offs for RQ3 was complex and should take sufficient resources for its validation from the side of practitioners. The possible mitigation is to follow the proposed verification strategy described further.

The main threat to internal validity associated with our approach to consider all quality attributes of equal importance as well as architectural tactics of equal value. Therefore, we do not weigh the magnitude of the trade-offs between quality attributes and the scale of the consequences of AT implementation. It can be mitigated by in-depth research of each AT with the study of the specific contexts, leaving generalizability behind.

The main threat to construct validity is our focus on commonly reported quality attributes in the literature, which we implicitly assume to be correlated with their generalizability and the need to include these attributes in a common quality model for the domain of ML-enabled software. Clearly, some of the less commonly reported attributes might still be important in particular domains and use cases. To mitigate this threat it is possible to run a separate study of such attributes.

5.5. Implications for practitioners

Our study has several implications for practitioners and researchers. Considering practitioners, our findings can be used in a checklist-like manner during the system requirements and design stages. During requirements elicitation, our quality model can guide practitioners in identifying relevant non-functional requirements for the overall system that then need to be addressed by the system architecture. The list of architectural tactics provides them with insights into how to achieve those quality attributes architecturally. We especially highlight our description of context and examples we provide for each tactic, which allows practitioners to match the tactics to their particular context at hand. Finally, the table of trade-offs raises awareness of possible unintended consequences of decisions made. These insights can be especially valuable for start-ups and SMEs with limited resources for hiring ML engineering experts.

Our findings are also informative for the emerging area of *MLOps*, which focuses on the integration of DevOps principles and practices into the development and maintenance of machine learning systems [235]. While a dedicated literature study of quality attributes and tactics for MLOps is outside our scope, we identify the following potential applications of our findings in an MLOps context. First, the identified quality attributes can help align the engineering process with business goals and operational needs and contribute to more sustainable software development. For example, the consideration of maintainability at the design stage can guide the optimization of planned resources for further maintenance, support, and updates of the deployed system [222]. Second, treating training and deployed systems as aspects of a single complex software architecture along with appropriate tactics during the design phase, can improve operations associated with system retraining [25]. Finally, recognizing trade-offs can help allocate resources and prioritize tasks within the common workflow [236]. For example, if a team uses containerization to improve maintainability, our findings emphasize potential drawbacks for security, which need to be addressed specifically within the roles and responsibilities of the DevOps setting, for example, by actively planning and estimating the effort arising for a security team within the company [237].

5.6. Implications for researchers

We suggest several directions for future work within the existing architectural perspective. First, our list of trade-offs can be informative for researchers to develop new architectural tactics. Given that our numbers of

identified tactics per quality attribute are between one and three, there is a potential for new tactics that complement the existing ones to find a different "sweet spot" within the trade-offs, possibly addressing specific domains and application contexts. Second, more generally, while our study was broadly focused on ML-based systems, it would be worthwhile to conduct additional studies that explore the relevance and applicability of our findings in different domains (e.g., automotive vehicles, healthcare systems, etc.). Third, our study of trade-offs is based on literature sources; yet, the results could benefit from follow-up research that verifies and refines the resulting table of trade-offs based on insights in industrial settings.

While we conducted a first evaluation of our results with experts, there is room for further evaluating our findings in complementary ways, focusing on their real-world applicability in specific contexts. As part of future work, we propose to conduct such evaluations with empirical methods, such as case studies, controlled experiments, and surveys. As a promising direction, we highlight *action research* [238], which is dedicated to deploying solutions in a specific real-world context—for example, in our case, specific quality attributes and tactics in an industrial MLOps environment with multiple teams. In such an environment, we aim to investigate if the introduction of a quality model helps teams to allocate their priorities more efficiently, if the identified architectural tactics can be applied to systems of different natures and sizes and be generalizable to new contexts, and how identified trade-offs affect the decisions made by MLOps teams. Such an evaluation may shit the focus from our current architectural perspective to a more operational one, which can be especially relevant in the context of MLOps.

6. Conclusion

This work contributes to the methodology of building software architecture for ML-enabled systems from the perspective of quality, offering ways to define it and achieve it through common architectural tactics with a consideration of possible side effects. Our focus is on theory-building, as we systematically identified and synthesized information from 206 research papers.

There are several worthwhile directions for future work. First, while our contributions are informed and validated by empirical insights from published literature, additional validation, for example, through expert feedback, is possible. Second, the selection of literature for analysis can be expanded

by including gray literature. As a result, the quality model and proposed architectural tactics can be refined. Third, one can conduct complementary forms of evaluation highlighting the applicability of ouf findings in specific real-world contexts, through action research and other empirical methods. The emerging area of MLOps provides a particularly promising avenue for such evaluations. Finally, we suggest follow-up research to further investigate the role of quality aspects mentioned infrequently in the literature (e.g., portability) and to systematically study the impact of the identified tactics on all quality aspects.

7. Data Availability

A supplementary artifact[1] is available on the Figshare platform, containing all details about our data sources, search strategy, and data extraction strategy.

8. Acknowledgments

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, and VR. We are grateful to the experts presented at the Swedish Requirements Engineering Meeting (SiREN) 2023 and ML engineers from the Swedish AI software company for providing expertise on our findings.

References

[1] G. Hulten, Building Intelligent Systems: A Guide to Machine Learning Engineering, Apress, 2018.

[2] R. Monson-Haefel, 97 things every software architect should know: collective wisdom from the experts, O'Reilly Media, Inc., 2009.

[3] I. O. for Standardization, ISO/IEC 25010:2011, Systems and software engineering — Systems and software Quality Requirements and Evaluation — System and software quality models, Tech. rep., ISO (2011).

[4] I. O. for Standardization, ISO/IEC 42010:2011 - Systems and software engineering — Architecture description, Tech. rep., ISO (2011).

[1] Supplementary Artifact: https://figshare.com/s/57b4fa3f53caecd4a5b1

[5] I. O. for Standardization, ISO/IEC 25059:2023 Software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Quality model for AI systems, Tech. rep., ISO (2023).

[6] X. Wang, M. Miao, A framework for requirements specification of machine-learning systems, in: SEKE, 2022, pp. 7–12.

[7] M. U. Hassan, M. H. Rehmani, R. Kotagiri, J. Zhang, J. Chen, Differential privacy for renewable energy resources based smart metering, JPDC (2019).

[8] L. Wanganoo, V. K. Shukla, Real-time data monitoring in cold supply chain through NB-IoT, in: ICCCNT, 2020, pp. 1–6.

[9] L. Bass, P. Clements, R. Kazman, Software architecture in practice, Addison-Wesley, 2003.

[10] I. O. for Standardization, ISO/IEC 9126:2001 Software engineering - Product quality, Tech. rep., ISO (2001).

[11] L. Lundberg, J. Bosch, D. Häggander, P.-O. Bengtsson, Quality attributes in software architecture design, in: IASTED, 1999, pp. 353–362.

[12] A. Serban, J. Visser, Adapting software architectures to machine learning challenges, in: SANER, 2022, pp. 152–163.

[13] P. Santhanam, Quality management of machine learning systems, in: EDSMLS, 2020, pp. 1–13.

[14] J. Siebert, L. Joeckel, J. Heidrich, A. Trendowicz, K. Nakamichi, K. Ohashi, I. Namba, R. Yamamoto, M. Aoyama, Construction of a quality model for machine learning systems, SQ (2022).

[15] H. Kuwajima, H. Yasuoka, T. Nakae, Engineering problems in machine learning systems, ML (2020).

[16] B. Gezici, A. K. Tarhan, Systematic literature review on software quality for AI-based software, ESE 27 (3) (2022) 66.

[17] X. Franch, S. Martínez-Fernández, C. P. Ayala, C. Gómez, Architectural decisions in AI-based systems: An ontological view, in: QUATIC, 2022, pp. 18–27.

[18] M. Bhat, K. Shumaiev, U. Hohenstein, A. Biesdorf, F. Matthes, The evolution of architectural decision making as a key focus area of software architecture research: A semi-systematic literature study, in: ICSA, 2020, pp. 69–80.

[19] H. Muccini, K. Vaidhyanathan, Software architecture for ML-based systems: What exists and what lies ahead, in: WAIN, 2021, pp. 121–128.

[20] B. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering, Tech. rep., EBSE Technical Report (2007).

[21] K. M. Habibullah, G. Gay, J. Horkoff, Non-functional requirements for machine learning: Understanding current use and challenges among practitioners, RE (2023).

[22] S. Vojíř, T. Kliegr, Editable machine learning models? a rule-based framework for user studies of explainability, ADAC (2020).

[23] J. W. Drisko, T. Maschi, Content analysis, Oxford University Press, USA, 2016.

[24] S. K. Reed, A taxonomic analysis of abstraction, Perspectives on Psychological Science 11 (6) (2016) 817–837.

[25] S. Peldszus, H. Knopp, Y. Sens, T. Berger, Towards ML-integration and training patterns for AI-enabled systems, in: International Conference on Bridging the Gap between AI and Reality, Springer, 2023, pp. 434–452.

[26] A. Vogelsang, M. Borg, Requirements engineering for machine learning: Perspectives from data scientists, in: REW, 2019, pp. 245–251.

[27] C. Kästner, E. Kang, Teaching software engineering for AI-enabled systems, in: ICSE, 2020, pp. 45–48.

[28] M. A. Ağca, S. Faye, D. Khadraoui, A survey on trusted distributed artificial intelligence, ECSA (2022).

[29] H. Liu, S. Eksmo, J. Risberg, R. Hebig, Emerging and changing tasks in the development process for machine learning systems, in: ICSSP, 2020, pp. 125–134.

[30] P. Haindl, T. Hoch, J. Dominguez, J. Aperribai, N. K. Ure, M. Tunçel, Quality characteristics of a software platform for human- AI teaming in smart manufacturing, in: QUATIC, 2022, pp. 3–17.

[31] F. Ishikawa, N. Yoshioka, How do engineers perceive difficulties in engineering of machine-learning systems?-questionnaire survey, in: CESI, 2019, pp. 2–9.

[32] A. Serban, K. van der Blom, H. Hoos, J. Visser, Adoption and effects

of software engineering best practices in machine learning, in: ESEM, 2020, pp. 1–12.

[33] Z. Wan, X. Xia, D. Lo, G. C. Murphy, How does machine learning change software development practices?, TSE (2019).

[34] R. H. Yap, Towards certifying trustworthy machine learning systems, in: TAILOR, 2021, pp. 77–82.

[35] I. Ozkaya, What is really different in engineering AI-enabled systems?, IEEE Softw. (2020).

[36] J. M. Zhang, M. Harman, L. Ma, Y. Liu, Machine learning testing: Survey, landscapes and horizons, TSE (2020).

[37] H.-L. Truong, Coordination-aware assurance for end-to-end machine learning systems: the R3E approach, in: AI Assurance, 2023, pp. 339–367.

[38] H. Kuwajima, F. Ishikawa, Adapting SQuaRE for quality assessment of artificial intelligence systems, in: ISSREW, 2019, pp. 13–18.

[39] J. Horkoff, Non-functional requirements for machine learning: Challenges and new directions, in: RE, 2019, pp. 386–391.

[40] Q. Chen, Y. Gong, Y. Lu, J. Tang, Classifying and measuring the service quality of AI chatbot in frontline service, Journal of Business Research (2022).

[41] A. Poth, B. Meyer, P. Schlicht, A. Riel, Quality assurance for machine learning–an approach to function and system safeguarding, in: QRS, 2020, pp. 22–29.

[42] H. Challa, N. Niu, R. Johnson, Faulty requirements made valuable: On the role of data quality in deep learning, in: AIRE, 2020, pp. 61–69.

[43] A. Chakraborty, R. Bagavathi, U. Tomer, A comprehensive decomposition towards the facets of quality in IoT, in: ICOSEC, 2020, pp. 759–764.

[44] A. Perera, A. Aleti, C. Tantithamthavorn, J. Jiarpakdee, B. Turhan, L. Kuhn, K. Walker, Search-based fairness testing for regression-based machine learning systems, ESE (2022).

[45] K. Nakamichi, K. Ohashi, I. Namba, R. Yamamoto, M. Aoyama, L. Joeckel, J. Siebert, J. Heidrich, Requirements-driven method to determine quality characteristics and measurements for machine learning software and its evaluation, in: RE, 2020, pp. 260–270.

[46] L. E. Lwakatare, A. Raj, I. Crnkovic, J. Bosch, H. H. Olsson, Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions, Inf. Softw. Technol . (2020).

[47] A. L. Smith, R. Clifford, Quality characteristics of artificially intelligent systems, in: IWESQ APSEC, 2020, pp. 1–6.

[48] H. Yokoyama, Machine learning system architectural pattern for improving operational stability, in: ICSA, 2019, pp. 267–274.

[49] H. Barzamini, M. Shahzad, H. Alhoori, M. Rahimi, A multi-level semantic web for hard-to-specify domain concept, pedestrian, in ML-based software, RE (2022).

[50] S. Khan, S. Tsutsumi, T. Yairi, S. Nakasuka, Robustness of AI-based prognostic and systems health management, Annu. Rev. Control. (2021).

[51] N. Balasubramaniam, M. Kauppinen, K. Hiekkanen, S. Kujala, Transparency and explainability of AI systems: ethical guidelines in practice, in: REFSQ, 2022, pp. 3–18.

[52] L. M. Cysneiros, J. C. S. do Prado Leite, Non-functional requirements orienting the development of socially responsible software, in: CAiSE, 2020, pp. 335–342.

[53] H. Washizaki, H. Uchida, F. Khomh, Y.-G. Guéhéneuc, Studying software engineering patterns for designing machine learning systems, in: IWESEP, 2019, pp. 49–495.

[54] K. Ahmad, M. Abdelrazek, C. Arora, M. Bano, J. Grundy, Requirements practices and gaps when engineering human-centered artificial intelligence systems, ASOC (2023).

[55] M. Felderer, R. Ramler, Quality assurance for AI-based systems: Overview and challenges (introduction to interactive session), in: SWQD, 2021, pp. 33–42.

[56] M. Felderer, B. Russo, F. Auer, On testing data-intensive software systems, C-CPS (2019).

[57] D. G. Arseniev, D. E. Baskakov, J. Kasurinen, V. P. Shkodyrev, A. Mergasov, Software engineering principles apply to artificial intelligence systems, in: CPS&C, 2021, pp. 151–158.

[58] M. M. Morovati, A. Nikanjam, F. Khomh, Z. M. Jiang, Bugs in machine learning-based systems: a faultload benchmark, ESE (2023).

[59] C. Tao, J. Gao, T. Wang, Testing and quality validation for AI software–perspectives, issues, and practices, JSS (2019).

[60] H.-M. Heyn, E. Knauss, P. Pelliccione, A compositional approach to creating architecture frameworks with an application to distributed AI systems, JSS (2023).

[61] J. Dodge, Q. V. Liao, Y. Zhang, R. K. Bellamy, C. Dugan, Explaining models: an empirical study of how explanations impact fairness judgment, in: IUI, 2019, pp. 275–285.

[62] G. Amaral, R. Guizzardi, G. Guizzardi, J. Mylopoulos, Ontology-based modeling and analysis of trustworthiness requirements: Preliminary results, in: ER, 2020, pp. 342–352.

[63] S. Picard, C. Chapdelaine, C. Cappi, L. Gardes, E. Jenn, B. Lefèvre, T. Soumarmon, Ensuring dataset quality for machine learning certification, in: ISSREW, 2020, pp. 275–282.

[64] S. V. Garbuk, Intellimetry as a way to ensure AI trustworthiness, in: IC-AIAI, 2018, pp. 27–30.

[65] F. Boenisch, V. Battis, N. Buchmann, M. Poikela, "i never thought about securing my machine learning systems": A study of security and privacy awareness of machine learning practitioners, in: MuC, 2021, pp. 520–546.

[66] S. Shi, Q. Wang, X. Chu, B. Li, Y. Qin, R. Liu, X. Zhao, Communication-efficient distributed deep learning with merged gradient sparsification on gpus, in: INFOCOM, IEEE, 2020, pp. 406–415.

[67] J. Rao, X. Bu, K. Wang, C.-Z. Xu, Self-adaptive provisioning of virtualized resources in cloud computing, in: SIGMETRICS, 2011, pp. 129–130.

[68] B. Zhao, M. Song, S. Liu, L. Sun, W. Jiang, H. Qian, X.-Y. Zhang, Y. Zhang, T. Jiang, Mosaicnet: A deep-learning-based multi-tile biomedical image stitching method, in: EMBC, IEEE, 2023, pp. 1–4.

[69] S. Noureen, M. Zubair, M. Ali, M. Q. Mehmood, Deep learning based sequence modeling for optical response retrieval of photonic nanostructures, in: IBCAST, IEEE, 2021, pp. 289–292.

[70] G. Drainakis, P. Pantazopoulos, K. V. Katsaros, V. Sourlas, A. Amditis, D. I. Kaklamani, From centralized to federated learning: Explor-

ing performance and end-to-end resource consumption, Computer Networks 225 (2023) 109657.

[71] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, W. Shi, Federated learning of predictive models from federated electronic health records, International journal of medical informatics 112 (2018) 59–67.

[72] A. P. Singh, S. Chaudhari, Embedded machine learning-based data reduction in application-specific constrained IoT networks, in: SIGAPP, 2020, pp. 747–753.

[73] M. H. ur Rehman, C. S. Liew, A. Abbas, P. P. Jayaraman, T. Y. Wah, S. U. Khan, Big data reduction methods: a survey, Data Science and Engineering 1 (2016) 265–284.

[74] A. M. Hussein, A. K. Idrees, R. Couturier, Distributed energy-efficient data reduction approach based on prediction and compression to reduce data transmission in IoT networks, International Journal of Communication Systems 35 (15) (2022) e5282.

[75] D. Petrelli, A.-S. Dadzie, V. Lanfranchi, Mediating between AI and highly specialized users, AI Magazine 30 (4) (2012) 95–95.

[76] M. Winter, P. Jackson, S. Fallahkhair, Gesture Me: A machine learning tool for designers to train gesture classifiers, in: CHIRA, Springer, 2023, pp. 336–352.

[77] M. Kröll, K. Burova-Keßler, AI and learning in the context of digital transformation, in: AHFE, Springer, 2021, pp. 36–43.

[78] F. Heimerl, S. Koch, H. Bosch, T. Ertl, Visual classifier training for text document retrieval, TVCG 18 (12) (2012) 2839–2848.

[79] F. Sperrle, M. El-Assady, G. Guo, R. Borgo, D. H. Chau, A. Endert, D. Keim, A survey of human-centered evaluations in human-centered machine learning, in: Computer Graphics Forum, Vol. 40, Wiley Online Library, 2021, pp. 543–568.

[80] O. Gómez-Carmona, D. Casado-Mansilla, D. López-de Ipiña, J. García-Zubia, Human-in-the-loop machine learning: Reconceptualizing the role of the user in interactive approaches, Internet of Things 25 (2024) 101048.

[81] G. A. Lewis, I. Ozkaya, X. Xu, Software architecture challenges for ML systems, in: ICSME, IEEE, 2021, pp. 634–638.

[82] T. Van Der Weide, D. Papadopoulos, O. Smirnov, M. Zielinski,

T. Van Kasteren, Versioning for end-to-end machine learning pipelines, in: DM-ML, 2017, pp. 1–9.

[83] S. J. Warnett, U. Zdun, Architectural design decisions for machine learning deployment, in: ICSA, IEEE, 2022, pp. 90–100.

[84] P. T. Rajendran, H. Espinoza, A. Delaborde, C. Mraidha, Human-in-the-loop learning methods toward safe DL-based autonomous systems: A review, in: SAFECOMP, Springer, 2021, pp. 251–264.

[85] P. Sanju, Enhancing intrusion detection in IoT systems: A hybrid metaheuristics-deep learning approach with ensemble of recurrent neural networks, JER 11 (4) (2023) 356–361.

[86] Q. Liu, L. Chen, H. Jiang, J. Wu, T. Wang, T. Peng, G. Wang, A collaborative deep learning microservice for backdoor defenses in industrial IoT networks, Ad Hoc Networks 124 (2022) 102727.

[87] F. Qu, J. Zhang, Z. Shao, S. Qi, An intrusion detection model based on deep belief network, in: ICNCC, 2017, pp. 97–101.

[88] S. Laqtib, K. E. Yassini, M. L. Hasnaoui, A deep learning methods for intrusion detection systems based machine learning in MANET, in: SCA, 2019, pp. 1–8.

[89] S. M. M. Rashid, M. Toufikuzzaman, M. S. Hossain, A deep learning based semi-supervised network intrusion detection system robust to adversarial attacks, in: NSysS, 2023, pp. 25–34.

[90] H. El Balbali, A. Abou El Kalam, AI-driven big data quality improvement for efficient threat detection in agricultural IoT systems, in: AI2SD, Springer, 2023, pp. 39–47.

[91] G. McGraw, Security engineering for machine learning (keynote), in: SIGSOFT, 2020, pp. 2–2.

[92] W. Bekri, R. Jmal, L. C. Fourati, Secure and trustworthiness IoT systems: investigations and literature review, TS (2024) 1–36.

[93] Y.-H. Wu, X.-H. Huang, J.-X. Liu, L. Chang, A big data encryption method based on lorenz and feistel structures, in: ICCEAI, IEEE, 2022, pp. 1–5.

[94] M. Kantarcioglu, F. Shaon, Securing big data in the age of AI, in: TPS-ISA, IEEE, 2019, pp. 218–220.

[95] Y. Aono, T. Hayashi, L. Wang, S. Moriai, et al., Privacy-preserving deep learning via additively homomorphic encryption, TIFS 13 (5) (2017) 1333–1345.

[96] M. Kim, O. Günlü, R. F. Schaefer, Federated learning with local differential privacy: Trade-offs between privacy, utility, and communication, in: ICASSP, IEEE, 2021, pp. 2650–2654.

[97] J. Zhou, Z. Su, J. Ni, Y. Wang, Y. Pan, R. Xing, Personalized privacy-preserving federated learning: Optimized trade-off between utility and privacy, in: GLOBECOM, IEEE, 2022, pp. 4872–4877.

[98] H. Kaur, V. Rani, M. Kumar, M. Sachdeva, A. Mittal, K. Kumar, Federated learning: a comprehensive review of recent advances and applications, Multimedia Tools and Applications (2023) 1–24.

[99] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, Y. Gao, A survey on federated learning, KBS 216 (2021) 106775.

[100] M. M. Rovnyagin, A. S. Hrapov, A. V. Guminskaia, A. P. Orlov, ML-based heterogeneous container orchestration architecture, in: EICon-Rus, IEEE, 2020, pp. 477–481.

[101] A. B. Kolltveit, J. Li, Operationalizing machine learning models: A systematic literature review, in: SE4RAI, 2022, pp. 1–8.

[102] M. Openja, F. Majidi, F. Khomh, B. Chembakottu, H. Li, Studying the practices of deploying machine learning projects on docker, in: EASE, 2022, pp. 190–200.

[103] J. González-Abad, Á. López García, V. Y. Kozlov, A container-based workflow for distributed training of deep learning algorithms in HPC clusters, Cl. Comp. 26 (5) (2023) 2815–2834.

[104] H. B. Braiek, F. Khomh, On testing machine learning programs, JSS 164 (2020) 110542.

[105] S. Singaravel, J. Suykens, P. Geyer, Deep-learning neural-network architectures and methods: Using component-based models in building-design energy prediction, Adv. Eng. Inform. (2018).

[106] J. Wonsil, J. Sullivan, M. Seltzer, A. Pocock, Integrated reproducibility with self-describing machine learning models, in: CRR, 2023, pp. 1–14.

[107] N. Shadab, A. Salado, Towards an interface description template for reusing AI-enabled systems, in: SMC, IEEE, 2020, pp. 2893–2900.

[108] P. Geyer, S. Singaravel, Component-based machine learning for performance prediction in building design, Applied energy 228 (2018) 1439–1453.

[109] N. Naydenov, S. Ruseva, Combining container orchestration and ma-

chine learning in the cloud: A systematic mapping study, in: INFOTEH, IEEE, 2022, pp. 1–6.

[110] S. Joshi, B. Hasan, R. Brindha, Optimal declarative orchestration of full lifecycle of machine learning models for cloud native, in: ICAAIC, IEEE, 2024, pp. 578–582.

[111] V. U. Gongane, M. V. Munot, A. D. Anuse, A survey of explainable AI techniques for detection of fake news and hate speech on social media platforms, JCS (2024) 1–37.

[112] D. Minh, H. X. Wang, Y. F. Li, T. N. Nguyen, Explainable artificial intelligence: a comprehensive review, Artificial Intelligence Review (2022) 1–66.

[113] J. Maan, Deep learning-driven explainable AI using generative adversarial network (GAN), in: INDICON, IEEE, 2022, pp. 1–5.

[114] K. Höhlein, M. Kern, T. Hewson, R. Westermann, A comparative study of convolutional neural network models for wind field downscaling, Meteorological Applications 27 (6) (2020) e1961.

[115] D. Gaspar, P. Silva, C. Silva, Explainable AI for intrusion detection systems: Lime and shap applicability on multi-layer perceptron, ICE/ITMC (2024).

[116] H. Saadatfar, Z. Kiani-Zadegan, B. Ghahremani-Nezhad, US-LIME: Increasing fidelity in LIME using uncertainty sampling on tabular data, Neurocomputing 597 (2024) 127969.

[117] N. Burkart, M. F. Huber, A survey on the explainability of supervised machine learning, JAIR 70 (2021) 245–317.

[118] P. E. Love, W. Fang, J. Matthews, S. Porter, H. Luo, L. Ding, Explainable artificial intelligence (XAI): Precepts, models, and opportunities for research in construction, AEI 57 (2023) 102024.

[119] R. Moraffah, M. Karami, R. Guo, A. Raglin, H. Liu, Causal interpretability for machine learning-problems, methods and evaluation, SIGKDD 22 (1) (2020) 18–33.

[120] C. P. Vieira, L. A. Digiampietri, Machine learning post-hoc interpretability: a systematic mapping study, in: SBSI, ACM, 2022, pp. 1–8.

[121] E. Soares, P. P. Angelov, B. Costa, M. P. G. Castro, S. Nageshrao, D. Filev, Explaining deep learning models through rule-based approximation and visualization, TFS (2020).

[122] D. Rajapaksha, C. Bergmeir, W. Buntine, LoRMIkA: Local rule-based model interpretability with k-optimal associations, Information Sciences 540 (2020) 221–241.

[123] Y. Rimal, N. Sharma, A. Alsadoon, The accuracy of machine learning models relies on hyperparameter tuning: student result classification using random forest, randomized search, grid search, bayesian, genetic, and optuna algorithms, Multimedia Tools and Applications (2024) 1–16.

[124] M. Daviran, A. Maghsoudi, R. Ghezelbash, B. Pradhan, A new strategy for spatial predictive mapping of mineral prospectivity: Automated hyperparameter tuning of random forest approach, Computers & Geosciences 148 (2021) 104688.

[125] A. L. C. Ottoni, A. M. Souza, M. S. Novo, Automated hyperparameter tuning for crack lee2021landscapeimage classification with deep learning, Soft Computing 27 (23) (2023) 18383–18402.

[126] P. Kerschke, H. H. Hoos, F. Neumann, H. Trautmann, Automated algorithm selection: Survey and perspectives, Evolutionary computation 27 (1) (2019) 3–45.

[127] N. Pise, P. Kulkarni, Algorithm selection for classification problems, in: SAI, IEEE, 2016, pp. 203–211.

[128] H. H. Rashidi, N. Tran, S. Albahra, L. T. Dang, Machine learning in health care and laboratory medicine: General overview of supervised learning and Auto-ML, International Journal of Laboratory Hematology 43 (2021) 15–22.

[129] F. Deeba, S. R. Patil, Utilization of machine learning algorithms for prediction of diseases, in: i-PACT, IEEE, 2021, pp. 1–7.

[130] M. Alissa, K. Sim, E. Hart, A feature-free approach to automated algorithm selection, in: GECCO, Wiley, 2023, pp. 9–10.

[131] M. S. A. Lee, J. Singh, The landscape and gaps in open source fairness toolkits, in: CHI, 2021, pp. 1–13.

[132] C. Ferrara, G. Sellitto, F. Ferrucci, F. Palomba, A. De Lucia, Fairness-aware machine learning engineering: how far are we?, ESE 29 (1) (2024) 9.

[133] A. Agarwal, H. Agarwal, A seven-layer model with checklists for standardising fairness assessment throughout the AI lifecycle, AI and Ethics (2023) 1–16.

[134] A. Castelnovo, R. Crupi, G. Greco, D. Regoli, I. G. Penco, A. C. Cosentini, A clarification of the nuances in the fairness metrics landscape, Scientific Reports 12 (1) (2022) 4209.

[135] J. Zhang, Y. Shu, H. Yu, Fairness in design: A framework for facilitating ethical artificial intelligence designs, IJCS 7 (1) (2023) 32–39.

[136] S. Siddiqi, R. Kern, M. Boehm, Saga: A scalable framework for optimizing data cleaning pipelines for machine learning applications, MoD 1 (3) (2023) 1–26.

[137] G. Canbek, S. Sagiroglu, T. T. Temizel, New techniques in profiling big datasets for machine learning with a concise review of android mobile malware datasets, in: IBIGDELFT, 2018, pp. 117–121.

[138] F. Mostafa, L. Tao, W. Yu, An effective architecture of digital twin system to support human decision making and AI-driven autonomy, CCPE 33 (19) (2021) e6111.

[139] I. Logothetis, S. Barnett, L. Hoon, S. Thudumu, J. Mathew, C. Luckhoff, G. O'Reilly, D. Collard, R. Vasa, K. Mouzakis, et al., Pims: A pre-ML labelling tool, in: e-Science, IEEE, 2022, pp. 431–432.

[140] R. R. Pansara, B. Y. Kasula, A. B. Bhatia, P. Whig, Enhancing sustainable development through machine learning-driven master data management, in: International Conference on Sustainable Development through Machine Learning, AI and IoT, Springer, 2024, pp. 332–341.

[141] R. Gawhade, L. R. Bohara, J. Mathew, P. Bari, Computerized data-preprocessing to improve data quality, in: ICPC2T, 2022, pp. 1–6.

[142] M. Ramkumar, K. Malathi, K. Pavithra, Optimizing machine learning model accuracy via OBNT algorithm: Advanced data preprocessing technique, in: ICSES, IEEE, 2023, pp. 1–6.

[143] L. Santos, L. Ferreira, Atlantic—automated data preprocessing framework for supervised machine learning, Software Impacts 17 (2023) 100532.

[144] M. Bilal, G. Ali, M. W. Iqbal, M. Anwar, M. S. A. Malik, R. A. Kadir, Auto-prep: efficient and automated data preprocessing pipeline, IEEE Access 10 (2022) 107764–107784.

[145] O. Nassef, W. Sun, H. Purmehdi, M. Tatipamula, T. Mahmoodi, A survey: Distributed machine learning for 5G and beyond, Computer Networks 207 (2022) 108820.

[146] H.-P. Cheng, P. Yu, H. Hu, S. Zawad, F. Yan, S. Li, H. Li, Y. Chen, Towards decentralized deep learning with differential privacy, in: ICCC, Springer, 2019, pp. 130–145.

[147] F. Zerka, V. Urovi, F. Bottari, R. T. Leijenaar, S. Walsh, H. Gabrani-Juma, M. Gueuning, A. Vaidyanathan, W. Vos, M. Occhipinti, et al., Privacy preserving distributed learning classifiers–sequential learning with small sets of data, Computers in Biology and Medicine 136 (2021) 104716.

[148] B. Guijarro-Berdiñas, S. Fernandez-Lorenzo, N. Sánchez-Maroño, O. Fontenla-Romero, A privacy-preserving distributed and incremental learning method for intrusion detection, in: ICANN, Springer, 2011, pp. 415–421.

[149] N. Mandela, I. Alam, A. Amudha, D. Priyanka, D. K. Singh, et al., Enabling scalable applications with intelligent distributed data processing, in: INCOFT, IEEE, 2023, pp. 1–7.

[150] A. Tuladhar, D. Rajashekar, N. D. Forkert, Distributed learning in healthcare, Trends of Artificial Intelligence and Big Data for E-Health (2023) 183–212.

[151] D. Fan, Y. Wu, X. Li, On the fairness of swarm learning in skin lesion classification, in: MICCAI, Springer, 2021, pp. 120–129.

[152] T. Lane, C. E. Brodley, Temporal sequence learning and data reduction for anomaly detection, TISSEC 2 (3) (2019) 295–331.

[153] M. Tomei, A. Schwing, S. Narayanasamy, R. Kumar, Sensor training data reduction for autonomous vehicles, in: MOBICOM, 2019, pp. 45–50.

[154] M. Z. A. Bhuiyan, T. Wang, A. Zaman, G. Wang, Data reduction through decision-making based on event-sensitivity in IoT-enabled event monitoring, in: HPCC, IEEE, 2019, pp. 2039–2044.

[155] S. Shen, T. Zhu, D. Wu, W. Wang, W. Zhou, From distributed machine learning to federated learning: In the view of data privacy and security, CCPE 34 (16) (2022) e6002.

[156] H. Jeong, T.-M. Chung, Security and privacy issues and solutions in federated learning for digital healthcare, in: FDSE, Springer, 2022, pp. 316–331.

[157] G. K. Jagarlamudi, A. Yazdinejad, R. M. Parizi, S. Pouriyeh, Exploring

privacy measurement in federated learning, The Journal of Supercomputing (2023) 1–41.

[158] S. Shin, M. Boyapati, K. Suo, K. Kang, J. Son, An empirical analysis of image augmentation against model inversion attack in federated learning, Cluster Computing 26 (1) (2023) 349–366.

[159] L. Lyu, H. Yu, X. Ma, C. Chen, L. Sun, J. Zhao, Q. Yang, S. Y. Philip, Privacy and robustness in federated learning: Attacks and defenses, IEEE transactions on neural networks and learning systems (2022).

[160] M. Kirienko, M. Sollini, G. Ninatti, D. Loiacono, E. Giacomello, N. Gozzi, F. Amigoni, L. Mainardi, P. L. Lanzi, A. Chiti, Distributed learning: a reliable privacy-preserving strategy to change multicenter collaborations using AI, EJNMMI 48 (2021) 3791–3804.

[161] F. Sattler, K.-R. Müller, T. Wiegand, W. Samek, On the byzantine robustness of clustered federated learning, in: ICASSP, IEEE, 2020, pp. 8861–8865.

[162] H. Lycklama, L. Burkhalter, A. Viand, N. Küchler, A. Hithnawi, RoFL: Robustness of secure federated learning, in: SP, IEEE, 2023, pp. 453–476.

[163] X. Gu, Z. Tianqing, J. Li, T. Zhang, W. Ren, K.-K. R. Choo, Privacy, accuracy, and model fairness trade-offs in federated learning, Computers & Security 122 (2022) 102907.

[164] T. N. Nguyen, R. Choo, Human-in-the-loop XAI-enabled vulnerability detection, investigation, and mitigation, in: ASE, IEEE, 2021, pp. 1210–1212.

[165] M. L. Jones, E. Kaufman, E. Edenberg, AI and the ethics of automating consent, SP 16 (3) (2018) 64–72.

[166] S. Jena, S. Sundarrajan, A. Meena, B. Chandavarkar, Human-in-the-loop control and security for intelligent cyber-physical systems (CPSs) and IoT, in: ICDSIAI, Springer, 2022, pp. 393–403.

[167] D. Xin, L. Ma, J. Liu, S. Macke, S. Song, A. Parameswaran, Accelerating human-in-the-loop machine learning: Challenges and opportunities, in: DEEM, 2018, pp. 1–4.

[168] Y. Kang, Y.-W. Chiu, M.-Y. Lin, F.-Y. Su, S.-T. Huang, Towards model-informed precision dosing with expert-in-the-loop machine learning, in: IRI, IEEE, 2021, pp. 342–347.

[169] D. M. Rodríguez, M. P. Cuéllar, D. P. Morales, Concept logic trees: enabling user interaction for transparent image classification and human-in-the-loop learning, AI (2024) 1–13.

[170] N. Zhang, R. Bahsoon, N. Tziritas, G. Theodoropoulos, Explainable human-in-the-loop dynamic data-driven digital twins, in: DDDAS, Springer, 2022, pp. 233–243.

[171] J. Liu, Human-in-the-loop ethical AI for care robots and confucian virtue ethics, in: ICSR, Springer, 2022, pp. 674–688.

[172] S. Kalananthan, A. Kichutkin, Z. Shang, A. Strausz, F. J. S. Bautiste, M. El-Assady, Mindset: A bias-detection interface using a visual human-in-the-loop workflow, in: ECAI, Springer, 2023, pp. 93–105.

[173] B. Ghai, K. Mueller, D-bias: A causality-based human-in-the-loop system for tackling algorithmic bias, TVCG 29 (1) (2022) 473–482.

[174] M. Priestley, F. O'donnell, E. Simperl, A survey of data quality requirements that matter in ML development pipelines, JDIQ 15 (2) (2023) 1–39.

[175] J. Jakubik, M. Vössing, N. Kühl, J. Walk, G. Satzger, Data-centric artificial intelligence, BISE (2024) 1–9.

[176] M. H. N. Yousefi, V. Degeler, A. Lazovik, Empowering machine learning development with service-oriented computing principles, in: SummerSOC, Springer, 2023, pp. 24–44.

[177] S. Tsimenidis, T. Lagkas, K. Rantos, Deep learning in IoT intrusion detection, Journal of network and systems management 30 (1) (2022) 8.

[178] R. Devendiran, A. V. Turukmane, Dugat-LSTM: Deep learning based network intrusion detection system using chaotic optimization strategy, Expert Systems with Applications 245 (2024) 123027.

[179] B. Lampe, W. Meng, A survey of deep learning-based intrusion detection in automotive applications, Expert Systems with Applications 221 (2023) 119771.

[180] M. Pawlicki, A. Pawlicka, M. Śrutek, R. Kozik, M. Choraś, Interpreting intrusions-the role of explainability in AI-based intrusion detection systems, in: CORES, Springer, 2023, pp. 45–53.

[181] C. Shand, R. Fong, U. Butt, How explainable artificial intelligence (XAI) models can be used within intrusion detection systems (IDS) to

enhance an analyst's trust and understanding, in: ICGS3, Springer, 2023, pp. 321–342.

[182] S. Aljawarneh, M. B. Yassein, W. A. Talafha, A multithreaded programming approach for multimedia big data: encryption system, MTA 77 (2018) 10997–11016.

[183] D. Weng, Performance and energy evaluation of lightweight cryptography for small IoT devices, in: UEMCON, IEEE, 2023, pp. 289–295.

[184] R. Cantoro, N. I. Deligiannis, M. S. Reorda, M. Traiola, E. Valea, Evaluating data encryption effects on the resilience of an artificial neural network, in: DFT, IEEE, 2020, pp. 1–4.

[185] C.-J. Wang, P.-P. Li, X.-Y. Zhou, N. Liu, Privacy-preserving breast cancer prediction via inner-product functional encryption, in: ICCC, IEEE, 2021, pp. 539–543.

[186] G. Gupta, K. Lakhwani, An enhanced approach to improve the encryption of big data using intelligent classification technique, Multimedia Tools and Applications 81 (18) (2022) 25171–25204.

[187] M. M. Rovnyagin, K. V. Timofeev, A. A. Elenkin, V. A. Shipugin, Cloud computing architecture for high-volume ML-based solutions, in: EIConRus, IEEE, 2019, pp. 315–318.

[188] C. Figueroa, T. Knowles, V. Kukreja, C.-H. Lung, IoT management with container orchestration, in: ICEIB, IEEE, 2023, pp. 49–54.

[189] N. Joraviya, B. N. Gohil, U. P. Rao, DL-HIDS: deep learning-based host intrusion detection system using system calls-to-image for containerized cloud environment, The Journal of Supercomputing (2024) 1–29.

[190] S. Arisdakessian, O. A. Wahab, A. Mourad, H. Otrok, Towards instant clustering approach for federated learning client selection, in: ICNC, IEEE, 2023, pp. 409–413.

[191] H. Siddiqui, F. Khendek, M. Toeroe, Microservices based architectures for IoT systems-state-of-the-art review, IoT (2023) 100854.

[192] H. S. Sarjoughian, F. Fallah, S. Saeidi, E. J. Yellig, Transforming discrete event models to machine learning models, in: WSC, IEEE, 2023, pp. 2662–2673.

[193] B. Heisele, P. Ho, T. Poggio, Face recognition with support vector machines: Global versus component-based approach, in: ICCV, Vol. 2, IEEE, 2011, pp. 688–694.

[194] N. B. Kumarakulasinghe, T. Blomberg, J. Liu, A. S. Leao, P. Papapetrou, Evaluating local interpretable model-agnostic explanations on clinical machine learning classification models, in: CBMS, IEEE, 2020, pp. 7–12.

[195] T. Mori, N. Uchihira, Balancing the trade-off between accuracy and interpretability in software defect prediction, ESE 24 (2019) 779–825.

[196] V. Bhargava, M. Couceiro, A. Napoli, LimeOut: an ensemble approach to improve process fairness, in: ECML PKDD, Springer, 2020, pp. 475–491.

[197] N. Burkart, M. Huber, P. Faller, Forcing interpretability for deep neural networks through rule-based regularization, in: ICMLA, IEEE, 2019, pp. 700–705.

[198] M. Soui, I. Gasmi, S. Smiti, K. Ghédira, Rule-based credit risk assessment model using multi-objective evolutionary algorithms, Expert systems with applications 126 (2019) 144–157.

[199] M. I. Rey, M. Galende, M. J. Fuente, G. Sainz-Palmero, Multi-objective based fuzzy rule based systems (FRBSs) for trade-off improvement in accuracy and interpretability: A rule relevance point of view., KBS 127 (2017) 67–84.

[200] L. Liao, H. Li, W. Shang, L. Ma, An empirical study of the impact of hyperparameter tuning and model optimization on the performance properties of deep neural networks, TOSEM 31 (3) (2022) 1–40.

[201] W. Romsaiyud, H. Schnoor, W. Hasselbring, Improving k-nearest neighbor pattern recognition models for privacy-preserving data analysis, in: Big Data, IEEE, 2019, pp. 5804–5813.

[202] L. Liu, J. Yu, Z. Ding, Adaptive and efficient GPU time sharing for hyperparameter tuning in cloud, in: ICPP, 2022, pp. 1–11.

[203] D. K. Jain, A. K. Dutta, E. Verdú, S. Alsubai, A. R. W. Sait, An automated hyperparameter tuned deep learning model enabled facial emotion recognition for autonomous vehicle drivers, Image and Vision Computing 133 (2023) 104659.

[204] Y. N. Kunang, S. Nurmaini, D. Stiawan, B. Y. Suprapto, Attack classification of an intrusion detection system using deep learning and hyperparameter optimization, JISA 58 (2021) 102804.

[205] L. Wu, G. Perin, S. Picek, I choose you: Automated hyperparameter

tuning for deep learning-based side-channel analysis, Trans. Emerg. Topics Comput. (2022).

[206] R. K. Batchu, H. Seetha, A generalized machine learning model for DDoS attacks detection using hybrid feature selection and hyperparameter tuning, Computer Networks 200 (2021) 108498.

[207] S. B. Feroz, N. Sharmin, M. S. Sevas, An empirical analysis of hyperparameter tuning impact on ensemble machine learning algorithm for earthquake damage prediction, Asian Journal of Civil Engineering (2024) 1–27.

[208] V. Perrone, M. Donini, M. B. Zafar, R. Schmucker, K. Kenthapadi, C. Archambeau, Fair bayesian optimization, in: AIES, 2021, pp. 854–863.

[209] X. Gao, J. Zhai, S. Ma, C. Shen, Y. Chen, Q. Wang, FairNeuron: improving deep neural network fairness with adversary games on selective neurons, in: ICSE, 2022, pp. 921–933.

[210] I. Dagan, R. Vainshtein, G. Katz, L. Rokach, Automated algorithm selection using meta-learning and pre-trained deep convolution neural networks, Information Fusion 105 (2024) 102210.

[211] J. Bossek, P. Kerschke, H. Trautmann, A multi-objective perspective on performance assessment and automated selection of single-objective optimization algorithms, ASC 88 (2020) 105901.

[212] S. Shahoud, M. Winter, H. Khalloof, C. Duepmeier, V. Hagenmeyer, An extended meta learning approach for automating model selection in big data environments using microservice and container virtualizationz technologies, IoT 16 (2021) 100432.

[213] R. Trajanov, S. Dimeski, M. Popovski, P. Korošec, T. Eftimov, Explainable landscape analysis in automated algorithm performance prediction, in: EvoStar, Springer, 2022, pp. 207–222.

[214] W. Hutiri, A. Y. Ding, F. Kawsar, A. Mathur, Tiny, always-on, and fragile: Bias propagation through design choices in on-device machine learning workflows, TSEM 32 (6) (2023) 1–37.

[215] M. Hort, Z. Chen, J. M. Zhang, M. Harman, F. Sarro, Bias mitigation for machine learning classifiers: A comprehensive survey, JRC (2023).

[216] R. Ghani, K. T. Rodolfa, P. Saleiro, S. Jesus, Addressing bias and fairness in machine learning: A practical guide and hands-on tutorial, in: SIGKDD, 2023, pp. 5779–5780.

[217] S. Jain, P. Kumar, Cost effective generic machine learning operation: A case study, in: ICDSNS, IEEE, 2023, pp. 1–6.

[218] Z. Chen, J. M. Zhang, F. Sarro, M. Harman, A comprehensive empirical study of bias mitigation methods for machine learning classifiers, TSEM (2023).

[219] M. Miceli, J. Posada, T. Yang, Studying up machine learning data: Why talk about bias when we mean power?, Human-Computer Interaction 6 (GROUP) (2022) 1–14.

[220] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, F. Herrera, A survey on data preprocessing for data stream mining: Current status and future directions, Neurocomputing 239 (2017) 39–57.

[221] M. C. Rendleman, J. M. Buatti, T. A. Braun, B. J. Smith, C. Nwakama, R. R. Beichel, B. Brown, T. L. Casavant, Machine learning with the TCGA-HNSC dataset: improving usability by addressing inconsistency, sparsity, and high-dimensionality, BMC bioinformatics 20 (2019) 1–9.

[222] K. Shivashankar, A. Martini, Maintainability challenges in ML: A systematic literature review, in: SEAA, IEEE, 2022, pp. 60–67.

[223] N. A. Hikal, M. Elgayar, Enhancing IoT botnets attack detection using machine learning-IDS and ensemble data preprocessing technique, in: ITAF, 2020, pp. 89–102.

[224] M. A. Bouke, A. Abdullah, An empirical study of pattern leakage impact during data preprocessing on machine learning-based intrusion detection models reliability, Expert Systems with Applications 230 (2023) 120715.

[225] C. V. G. Zelaya, Towards explaining the effects of data preprocessing on machine learning, in: ICDE, 2019, pp. 2086–2090.

[226] M. M. Basha, P. Kuppusamy, F-DDPT: An efficient fuzzy-based automated preprocessing technique to support explainability, in: ICCDN, Springer, 2022, pp. 283–296.

[227] Y. Sun, F. Haghighat, B. C. Fung, Trade-off between accuracy and fairness of data-driven building and indoor environment models: A comparative study of pre-processing methods, Energy (2022).

[228] H. S. Obaid, S. A. Dheyab, S. S. Sabry, The impact of data preprocessing techniques and dimensionality reduction on the accuracy of machine learning, in: IEMECON, IEEE, 2019, pp. 279–283.

[229] S. Sajid, B. M. von Zernichow, A. Soylu, D. Roman, Predictive data transformation suggestions in grafterizer using machine learning, in: MTSR, Springer, 2019, pp. 137–149.

[230] S. Oppold, M. Herschel, A system framework for personalized and transparent data-driven decisions, in: CAiSE, Springer, 2020, pp. 153–168.

[231] N. B. Ding, E. Mit, A framework of data quality assurance using machine learning, in: CITA, IEEE, 2023, pp. 88–93.

[232] E. Seo, H. Kim, T.-M. Chung, Profiling-based classification algorithms for security applications in Internet of Things, in: ICIOT, IEEE, 2019, pp. 138–146.

[233] W. Epperson, V. Gorantla, D. Moritz, A. Perer, Dead or alive: Continuous data profiling for interactive data science, IEEE Transactions on Visualization and Computer Graphics (2023).

[234] S. Tverdal, A. Goknil, P. Nguyen, E. J. Husom, S. Sen, J. Ruh, F. Flamigni, Edge-based data profiling and repair as a service for IoT, in: IoT, 2023, pp. 17–24.

[235] S. Alla, S. K. Adari, S. Alla, S. K. Adari, What is MLOps?, Beginning MLOps with MLFlow: Deploy Models in AWS SageMaker, Google Cloud, and Microsoft Azure (2021) 79–124.

[236] S. Barney, K. Petersen, M. Svahnberg, A. Aurum, H. Barney, Software quality trade-offs: A systematic map, Information and software technology 54 (7) (2012) 651–662.

[237] V. Mohan, L. B. Othmane, Secdevops: Is it a marketing buzzword?-mapping research on security in devops, in: 2016 11th international conference on availability, reliability and security (ARES), IEEE, 2016, pp. 542–547.

[238] M. Staron, Action research in software engineering, Springer, 2020.