CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Studying Software Architecture Design Challenges, Best Practices and Main Decisions for Machine Learning Systems

Software Architecture Design best practices, challenges, and main software architecture design decisions of Machine Learning Systems

Master's thesis in Computer science and engineering

ROGER NAZIR

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021

Master's thesis 2021

Studying Software Architecture Design Challenges, Best Practices and Main Decisions for Machine Learning Systems

Software Architecture Design best practices, challenges, and main software architecture design decisions of Machine Learning Systems

ROGER NAZIR

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2021

Studying Software Architecture Design Challenges, Best Practices and Main Decisions for Machine Learning Systems
Software Architecture Design best practices, challenges, and main software architecture design decisions of Machine Learning Systems
ROGER NAZIR

Supervisor: Patrizio Pelliccione, Department of Computer Science and Engineering
Examiner: Gregory Gay, Department of Computer Science and Engineering

Master's Thesis 2021
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Abstract:

Machine learning (ML) refers to statistical modeling techniques, which have recently sparked interest in the ML applications and service industries. The continuous usage of machine learning necessitates addressing software architecture (SA) design challenges and requires guidelines to overcome design issues. ML software system design in small and especially in big software engineering projects is a collaborative decision-making process in which software architect designers, researchers, and developers make design decisions. After considering various design alternatives, the development team handles design issues, examines the best design practices, and picks the main design decisions. In this paper, we provided the common challenges, best design practices, and major decisions for designing the software architecture of ML systems. The systematic literature approach (SLA), with snowballing, is used to extract academic papers from four databases. The inclusion/exclusion techniques helped to extract relevant articles according to research questions. Total 12 interviews were conducted from 9 countries across five continents with academic researchers and industrial professionals having a machine learning experience. Finally, SLR outcomes and interview results are mapped. The mapped and unmapped data are discussed in this study to get more insight into the software architecture design decision for machine learning systems.

Keywords: Machine Learning, Software Architecture Design, Common Design Challenges, Best Design Practices, Main Design Decisions

Acknowledgments:

I would like to offer my special thanks to supervisor Patrizio Pelliccione for his commitment and for providing consistent and accurate feedback throughout the thesis process. I am very grateful and express gratitude to my examiner, Gregory Gay, for the assistance, advice, and guidance in completing this research work. I pay the heartfelt gratitude to my family for their unconditional support during this journey. In the end, I want to acknowledge all practitioners who have taken the time for their assistance with the collection of my data.

Roger Nazir, Gothenburg, September 2021

Contents

List of Tables

List of Figures:

Abbreviations

ML Machines Learning

SA Software Architecture

SLR Systemic Literature Review

RQ Research Question

RQ1 Research Question One

RQ2 Research Question two

RQ3 Research Question three

Chapter 1

1
Introduction

1.1 Overview

Machine learning contributes to statistical modeling approaches, which have created recent interest in the marketplace of applications and services [1]. Software engineering research into the use of ML technologies has expanded enormously and generated a wide range of projects and publications, e.g. [2], [3], [4], [5], [6], [7], [8]. As a result, companies that produce software witnessed increasing customers in their software systems that require ML-based components and solutions [9]. ML solutions are used in several fields, including computer defense, computational biology, autonomous vehicles, robotics, and the Internet of Things (IoT). ML primarily depends on software engineering design techniques for its deployment and success. [2].

The architecture design of an ML software project has a significant impact on its performance. Designing the best software architecture design, on the other hand, is a very subjective process that takes a long time and is heavily influenced by the architect's understanding and the reliability of the requirement engineering. The architecture design understanding is generally not written because it is regarded as implicit knowledge by architects or other stakeholders, and this knowledge gradually fades away. It's particularly relevant when designing software or machine learning products that necessitate making decisions on various architecture design challenges, which often include choosing between various design alternatives that have varying effects on a collection of functional properties [10]. However, there is a systematic study analysis about how practitioners perceive and use ML design decisions in the architecture of ML systems and applications.

This research aims to assess the current state of the art in how teams/architects manage software architecture design decisions, including machine learning components. It's essential to require knowledge while making design decisions for developing ML systems. Researchers and experts are investigating best design practices in the software architecture design for ML systems and further about the ML system's complexity [2].

The software architecture design decision is defined as it's an essential part of software development. The various design decisions are made about processes, selection of tools, methods, and appropriate techniques throughout the software development life cycle. However, software architecture decisions are essential because many architecture decisions are made early in the software development life cycle and have a significant influence. The best design practices of ML systems encapsulate reusable solutions to solve common issues. The best design practices can provide recommendations to follow when designing the ML systems. On the other hand, the main design decisions are to find the best available design solutions from literature and practitioners' perspectives. The main design decisions can be the most appropriate/suitable design solution from other alternatives and carry more weight in developing a successful system.

Much work has been done on ML mathematics, computer science, identified mathematical challenges to develop ML systems. According to current findings [2], [3], [4], [5], [6], [7], [8], not much work is done to investigate the best design decisions for ML techniques, which are then classified and discussed. Thus, there is a gap in finding common software architecture (SA) design challenges, best design practices, and especially the main architecture design decisions for ML systems. Because of these concerns, the following questions arise; what are common SA design challenges in ML systems? What are the best practices in architecting and designing machine learning systems? What are the main software architectural design decisions for ML systems?

To find the answers to the questions, I compiled common software architecture design challenges, main design decisions, best practices in architecting and designing ML systems based on academic literature on four databases and practitioner's perspectives. The systematic literature review is applied to selected databases along with snowballing. The interview data are collected to get more knowledge for SA design decisions in ML systems. Both SLR and interview data are mapped in tables for all three research questions (RQs), and unmapped data are also available in tables.

However, ensuring the correctness of design decisions is not easy when architectural design planning occurs at an early stage, making it complicated. The developers also require a template recommendation to help by proposing appropriate design decisions [11]. This study would help to evaluate the SA design practices, challenges, and major architecture design decisions in the ML systems. For example, SLR and interview results findings state that the most common challenges are ML system's architecture design is data management, data pre-processing, and a lack of understanding of the company's needs. The most well-known problem in machine learning systems is selecting irrelevant models and architectures while making design decisions [12]. ML systems vary from software engineering systems because ML systems mainly depend upon data input to provide usable results. These types of concerns are beneficial while taking SA design decisions for ML systems. The results and discussion chapters explain similar challenges, best design practices, and main SA design decisions.

This paper is organized as follows: Initially, I discussed the background in Chapter 2. I explained the related work in Chapter 3. The methodology of the study is in Chapter 4. I discussed the results in chapter 5 and concluded with general observations and future work in chapter 7.

1.2 Significance of the study

This paper explored the software architecture design issues and their solutions. There is not much data collected for the challenges, best design practices, and especially the main design decision for the ML systems. According to my knowledge, no such paper was found which collected the data for the main architecture design decisions for ML systems. The best practices and challenges for SA design in one article can help the software developer understand the ML system’s designing difficulties. I hope this study can offer essential contributions in future research work for combined challenges, best/wrong practices, and the main software architectural design decisions for ML systems.

Chapter 2

2
Background

2.1 Statement of the problem

A software's architecture acts as a blueprint for the system. It provides an abstraction for managing system complexity and establishing a method for communication and coordination among components [10].

Machine learning is an artificial intelligence (AI) technology that allows computers to learn and improve from experience without being explicitly programmed automatically. Machine learning is concerned with creating computer programs that can access data and utilize it to learn independently. Machine learning systems predict output results by using historical data as input.

The process of conceptualizing software requirements and their translation into software implementation is known as software design decisions. A series of architectural design decisions influence the system's quality, performance, maintainability, and overall success. Failure to consider common design problems can compromise your system. The software architecture design process establishes a plan that considers the user requirements as challenges and identifies optimal solutions when developing the software. Software architecture designing decisions play a crucial role in guiding management practices during the growth and maintenance of software systems [5], [13]. Design practices are also formally recognized as architecture design decisions by encapsulating reusable responses to issues that frequently arise in the software architecture design of machine learning systems.

It is important to deal with poor architectural design choices that result in inadequate ML systems, which would be very expensive and impossible to correct later.

Chapter 3

3
Related Work

3.1 Review of the literature

The work in [1] reports a case study on software engineering for machine learning at Microsoft. The authors came up with the typical nine-staged process informed by prior experiences developing AI applications and data science tools in a paper. Moreover, in a questionnaire, the replies from 551 software engineers validated these challenges.

Hironori Washizaki [2] performed a systematic literature review on both academic and grey literature to collect the design patterns for ML systems. The authors used ten academic scholar papers and 25 grey documents and presented the preliminary findings of SLR to find design patterns in ML.

Another paper by Hironori Washizaki [14] focuses on product quality attributes, model quality attributes prediction, and ML pattern Pi. During their literature analysis, the authors discovered and studied 15 ML trends. Forty-three percent of respondents of surveys said the authors had reused previous solutions in the form of internal guidelines or trends. The study recommended that existing ML patterns be more widely used if promoted within the ML community.

Watanabe Yasuhiro's [7] survey extracted ML system's design patterns; the authors performed the systemic preliminary analysis of the literature on ML system's implementation. Their survey included practitioners and grey literature. Results from this study highlighted 33 design patterns of ML systems.

P. Avgeriou and U. Zdun [9] surveyed general architecture and design patterns and focused on object-oriented design. In a paper, the authors discussed; architectural patterns are a vital concept in software architecture. Finding and applying the appropriate architectural patterns in practice remains largely ad hoc and unsystematic. The authors proposed a pattern language that superset existing architectural pattern collections and categorizations [9].

In software development, architecture design decisions can provide reusable and recording solutions to fundamental design problems. Gang of Four (GoF) [3], the design paper presented the findings of a mapping survey of approximately 120 primary studies to provide an analysis of (GoF) style pattern science. The author's most active research areas tend to be pattern recognition and the impact of GoF trends on software quality attributes. [3]

Computer engineers generally use design trends to create complicated structures. The most active themes in design patterns are pattern creation, pattern mining, and pattern use. Art of design pattern paper [8] summarized research activities for researchers seeking to penetrate this field on design trends. The final supplementary thesis for related purposes only mentions the four style trends of the Gang of Four. It concludes that architecture trends can be divided into six separate subjects for study [8].

Other papers discussed the architecture design or design patterns for a limited domain, such as the multi-agent application system (MAS), which indicates the lack of overview for MAS patterns [4]. Likewise, IoT design Patterns [15], analyze many reported architectural IoT improvements and describe multi-dimensional design trends. The work [6] proposes security design patterns and explains the importance of understanding the security area necessary to adapt suitable design decisions for a particular problem context.

Wang Juguo [16] proposes a scenario-based architecture for artificial intelligence software reliability design. The authors divided the scenario into both parts that are environmental scenarios and structure scenarios. Both types of procedures are taken into account in the context. The software assesses and forecasts its quantitative reliability depending on the framework. So, the article presented the reliability of a pattern recognition system that could be evaluated and predicted with the design and scenario-based analysis to allow it to be eligible in safety-sensitive applications.

Mayer Ruben [17] emphasizes Deep Learning (DL); the papers focus on the scalability of DL systems that must continue to be enhanced to improve DL performance. The authors performed a detailed and comprehensive survey of challenges, methods, and resources for flexible DL on distributed facilities. These are DL infrastructures, parallel DL training approaches, multi-tenant resource planning, training, and model data management. The authors also reviewed and evaluated 11 existing open-source DL applications tools and studied the most used techniques. Finally, The authors underlined future developments in research into DL systems that need more research.

Wan Zhiyuan [18] conducted a mixture of 14 respondents from 26 countries on four continents in a mix of qualitative and quantitative studies to generate significant uncertainties between implementing machine learning systems and creating non-machine learning systems. Their analysis identifies major variations in different software engineering dimensions (e.g., requirements, architecture design, testing, and method) and job properties, e.g., diversity of skills, solving problems, and identifying variations. Based on their conclusions, the authors highlighted potential guidance for the study and provided practitioners the recommendations.

Henry Muccini [19] study concentrated on the former aspect of the spectrum, intending to highlight the various activities in the architecture of the ML-based software systems in the present scenario. To better define best practices of software-based architecture, the authors recognized four main fields for software architecture that require the attention of both ML and software practitioners. The authors focused our experience developing an ML-based software framework to solve the challenges of queuing in one of Italy's most prominent museums.

Yokoyama Haruki [20] considers the machine learning nature of closely coupled features, such as business logic coded from architecture and inference engines derived from data. Common machine learning systems with three-layer architectural patterns complicate the troubleshooting method. He suggested a new architecture pattern for machine learning systems that distinguish business logic and machine learning components to solve this problem. This architectural pattern allows operators to disassemble errors into a business logic component and a separate ML part. Thus, while the inflection engine has problems, the authors roll back the inference engine independently of the business logic. Through a case study example, he demonstrated how our architectural pattern would help solve problems more efficiently than common architecture in three different layers.

3.2 Summary of related work

From the above literature, most papers focused on the designing of object-oriented systems and particular domains. However, Hironori Washizaki [2] focused on finding design patterns, and the authors performed SLR on one database and conducted a survey in which a response was 1 percent only. In another article, Hironori Washizaki [15] focuses on product quality characteristics, model quality attribute prediction, and ML pattern Pi. So, both studies are not conveying the common challenges, best design practices, and main SA design decisions for ML systems which demonstrates the gap to address these concerns.

Mayer Ruben [17], focused on enhancing the performance of the DL-based systems. Concluding, to the best of my knowledge, there is not much data collecting, discussing, understanding, and classifying the SA design challenges, best design practices, and major architectural decisions on SA design for ML application systems. Most importantly, no paper-related work is found yet, which primarily discusses common design challenges, best practices, and the main architecture design decision for ML systems.

Chapter 4

4
Methods

4.1 Research question and/or Hypotheses

This study aims to find the common challenges, best design practices, and main software architecture design decisions of the ML systems. The major research questions of this study are:

1) What are the common software architecture design challenges in machine learning systems?
   I compiled the challenges of software architecture design in ML systems to discuss all common challenges in a single paper from available literature and conducted interviews with ML practitioners. This study can assist in minimizing the challenges of creating ML solutions.

2) What are the best practices in architecting and designing machine learning systems?
   The importance of finding the best practice in architecting and designing ML systems can help ML engineers and academic researchers follow the recommended practice that can provide accurate design solutions for their ML systems.

3) What are the main software architectural design decisions for machine learning systems?
   This analysis aims to investigate the main software architecture design decision between other alternatives for the software architecture of machine learning systems. According to my knowledge, no such paper collected the main architecture design decision of the machine learning systems. This study can offer some essential contributions for future research work and guide the main architectural design decision for ML engineers.

4.2 The Design – Methods and Procedures

I split the approach into three sections to find the answer to research questions: First, I Performed SLR to collect the data from the available literature. Secondly, I conducted interviews with some professionals to collect more data. Thirdly, I performed a comparison of SLR results with data collected from the interviews. Both result's mapping (comparison results) can be considered SLR outcomes evaluation and effectiveness of the results of common challenges, best design practices, and main architecture decisions for ML systems.

4.2.1 Systematic Literature Review

This section discussed the systemic literature review description and motivation of using all processes and selection of databases and extraction techniques to find primary studies.

4.2.1.1 Description and Motivation:

A systematic literature review is a way of reviewing and analyzing all existing findings relating to a specific research issue, current topic, or phenomena of interest [21], [22]. The most suitable approach for this study is SLR to extract the data to obtain the literature search list. Data gathering is based on research questions for determining the challenges and best design practices of SA designing ML systems. SLR is the best alternative because this technique allows for collecting and responding to relevant information on a given subject in line with the research questions. SLR can apply to qualitative and quantitative data information. So, I follow the mixed-method approach [23]. This method helps to produce quantitative and qualitative information from various research [24].

I followed the Kitchenham guidelines [12] to perform the systematic literature review. Kitchenham et al. [12] explained the SLR as analyzing, reviewing, and evaluating all available research material on a particular research issue or field. SLR also helps to plan the new research activities [25]. The data collected using SLR helped identify the design practices, challenges of architecture design, and ML system's main architecture decisions.

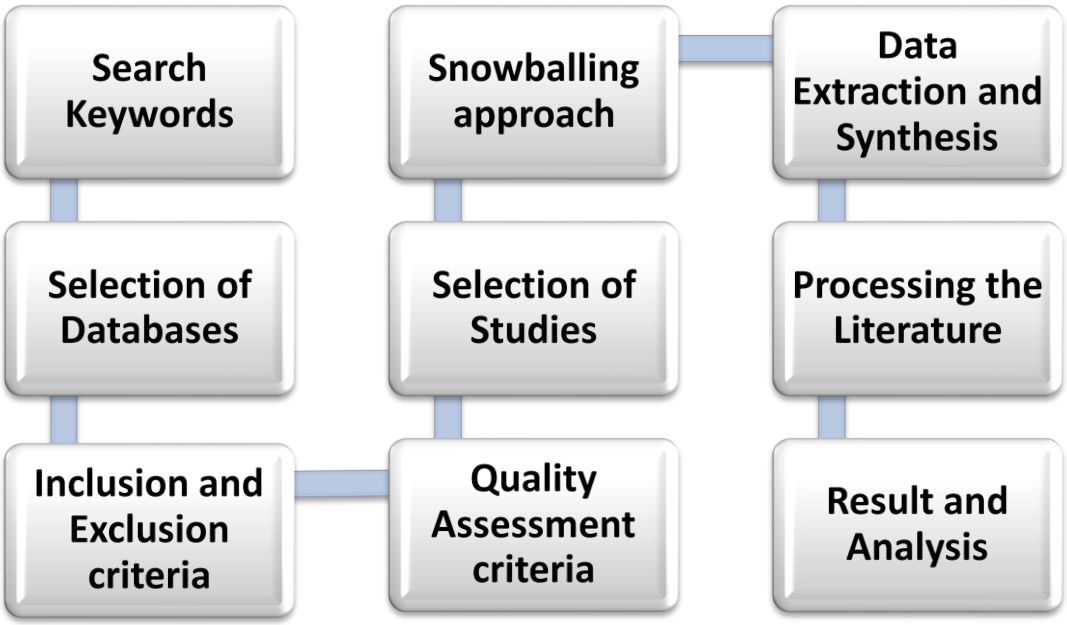

Figure 1 SLR Flow

4.2.1.2 Data Collection:

To gather the data through SLR, I followed the Kitchenham guidelines [12] to conduct a systematic literature review and obtain a list of papers from the literature search.

4.2.1.3 Selection of Database

Following are the databases which are selected in finding the primary studies to address research questions:

- IEEE Xplore
- ACM
- Scopus
- Web of Science

4.2.1.4 Search String:

I used the search string to retrieve papers that are related to research questions. The search string focused on software architecture, machine learning, challenges, best practices, and recommendation.

To obtain the primary studies, I conducted a database analysis. The search string applied on four databases collected all available papers till the end of 20 June 2021. I am going through four databases using the same search strings. The following search string is used in all four databases:

"("software architecture" AND ("machine learning" OR "AI" OR "deep learning" OR "artificial intelligence") AND (("challenge" OR "problem" OR "issue") OR ("best practice" OR "recommendation" OR "tactics")))".

It was important to point out that search string can find the following papers that I already mentioned in the related work section and used as control papers [1], [2], [14]. The papers gave us some confidence that the search string is well formulated.

To properly select those papers that focus on design activities relevant to the study, I define suitable inclusion and exclusion criteria, as described in the following.

4.2.1.6 Inclusion/Exclusion criteria:

The selection of studies for systematic review and meta-analysis is a time-consuming method. The duplication literature occurrence is expected. The selecting process is essential and narrows the criteria so that the inclusion and exclusion criteria govern the systemic review result's scope and effectiveness [26]. Exploiting inclusion and exclusion criteria can select the most relevant papers for SA designing best design practices, challenges, and major architecture design decisions for ML systems.

4.2.1.5.1 Inclusion Criteria:

I divided the inclusion criteria into two iterations. The first iteration is to apply the search string on the database and selected the paper according to the iterations. The selected articles in iteration one were further analyzed in iteration two, and their inclusion criteria are stated below:

Iteration 1:

- The articles should be written in English.
- The articles should be available in full text.
- First, reading the article title, then the abstract, and finally the conclusion. The initial understanding of the articles is learned. While reading each paper, the following key concepts were kept in mind for the paper's selection in Iteration one.
  i. "Software architecture/SA design decision for ML."
  ii. "Software architecture/SA best-practices/recommendation/tactics for ML."
  iii. "Software architecture/SA challenges/problem/issue for ML."
  iv. "Software architecture/SA main design decision for ML."

Iteration 2:

- Iteration 1 selected articles methodology text, discussion, and results focus any research question of the study.
- Further, the supervisor checked all selected papers found at the end of iteration two for verification before extraction results.

4.2.1.5.2 Exclusion Criteria:

As exclusion criteria, I considered the following steps:

- Literature is not available in English.
- The literature is not available in full text.
- Literature did not include the software architecture design decisions of machine learning, including best design practices and common challenges.
- Duplicate literature and the material present in the form of presentations and general discussion.
- The duplicate literature is found in two databases (selected only one).

4.2.1.7 Quality Assessment Criteria:

The quality standards for selected studies are represented in the following criteria. Quality characteristics apply to a selection of primary studies obtained through the database and snowballing. Every study's quality is determined by answering each question with "Yes," "Partially," or "No."

- Does the study identify the SA design challenges for the ML systems?
- Does the study report the best practices in architecting and designing ML systems?
- Does the study identify the main SA design decisions for the ML systems?

4.2.1.8 Snowballing Approach:

Since a database search could not yield all relevant results [27], I combined the snowballing method with a database search. Another reason is that database search strings cannot return all relevant studies. The various databases employ various interfaces and search strings that are limited to specific approaches in searching. It is challenging to construct a search string that returns all applicable studies since the keywords used are not standardized. Many

unrelated publications from database searches can also be found. Snowballing can be performed both backward and forward.

The principle of the forward search was introduced by Webster and Watson [28]. The process of checking references contained in keyword search results is known as backward reference search. By using keywords, you can find the most up-to-date literature. The phenomenon of dealing with forwarding reference search is known as forwarding search. The aim of forward snowballing is to discover new studies based on the citations of existing ones. The process of recognizing papers that referenced the original article or work since it was written. The knowledge base on the subject has expanded the scope of this.

Iteration [27] should be used for forward and backward snowballing. The most related research/study should be added to the initial set of tasks. The iteration of snowballing of newly discovered studies can be repeated until no further studies are detected. According to Wohlin [27], the starting list must consist of studies used in the SLR primary studies. The initial collection of papers should contain a variety of reports by various scholars and publishers.

4.2.1.9 Selected Papers:

Using the search strings, the total results in IEEE Xplore are 574 with applying the search string. ACM gave 5318 results, and similarly, other databases and snowballing combined results are 6494. After evaluating the relationship with research objectives and questions and applying filtering, inclusion/ exclusion criteria, I narrowed down and selected related papers. To choose or eliminate the articles, I studied all selected papers' introductions, methodologies, discussion/results, and conclusions.

Database	**Total**	**Selected**
IEEE Xplore	574	6
ACM	5318	8
Scopus	497	19
Web of Science	60	1
Snowballing	45	7
All	6494	41

Table 1 Selected results from Databases.

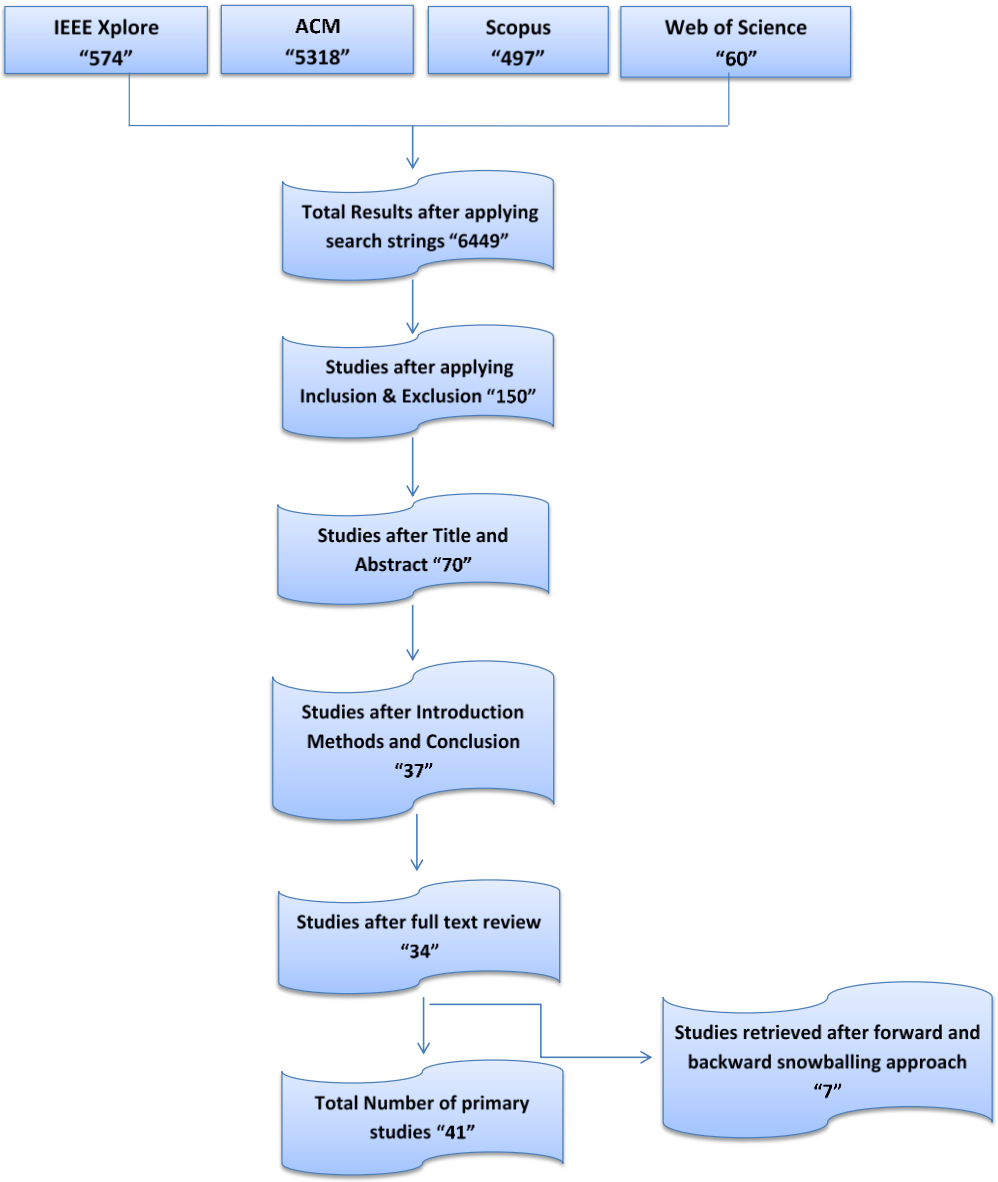

Figure 2 Primary Studies Selection Process

4.2.2 Survey

A survey is used to collect quantitative and qualitative data [29]. According to Floyd J. Fowler [30], the survey is the best tool for gathering data from practitioners through various organizations. The current study includes investigating the SA design challenges, designing best practices, and main architecture design decisions for ML systems, which is why a survey is selected as a research method.

Shull et al. [31], characterized the survey as identifying characteristics of a prominent individual population. In most surveys, the data are collected through interviews and questionnaires. I selected the interview approach for data collection.

4.2.2.1 Interview Data Collection

Interviews can be used in surveys as a means of data collection. According to G. Hackett [32], the most common data collection method in surveys is interviews and questionnaires. Interviews provide effective and thorough input on a particular subject. Open-ended questions are better suited for this study because researchers can dig deeper by addressing the question in depth and obtain more knowledge. Closed-ended questions are not suited in this study.

I created two pdf documents, the interview invitation letter and interview questionnaires, and both document's texts are presented in Appendix B. The supervisor confirmed both documentation's validation before using it. My supervisor and I sent interview invitations to several industry professionals, and academic researchers having primary work or focus on machine learning systems. I contacted 74 machine learning architects and machine learning engineers through LinkedIn, and nine industrial machine learning professionals agreed for the interview. The supervisor also contacted some machine learning experts, and three agreed to the interview. Because most interviewees resided in various time zones, the interview meeting time was coordinated through email. A total, 12 interviewees, participated in the interviews and answered interview questionnaires. Table 5 contains the statistics about the interviewees.

I conducted 12 interviews following Wohlin's [33] guidelines with professionals from various organizations as part of the survey for mapping with the SLR outcomes. To target professionals from other countries, I used the zoom feature to perform the online interviews and shared the zoom meeting links through an email. The interviews were recorded after the participant's permission except for one interviewee, and notes were taken with this interviewee. This information was stored in the system on Google Drive and shared with the supervisor for assurance of usefulness in the study. During the interview, I asked for their job title, and the interviewee's professions were ML academic researchers, ML architects, ML engineers/developers, and having machine learning systems knowledge.

The supervisor and myself contacted machine learning experts from all around the world, and 12 agreed to the interview, and the interviewees were in the following continents:

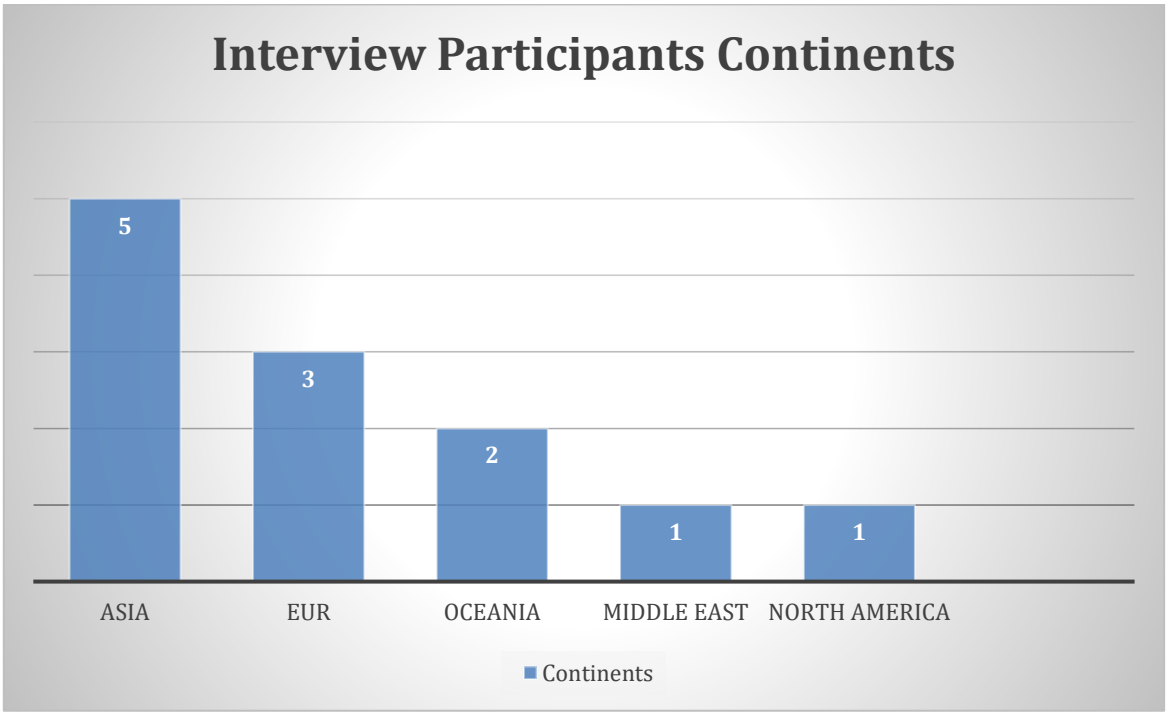

Figure 3 Interview Participants Continents

4.3 Comparison Validation:

Data are collected using SLR from literature and by conducting interviews with practitioners. The results extracted through SLR and interview outcomes are added in tables. Each research question of SLR is added in table 2, 3, and 4. Similarly, the interview outcomes of each research question are added in table 6, 5, and 8. Further, I compared each interview data according to common challenges, best design practices, and main software architecture design decisions and mapped it in tables 9, 10, and 11. Each interview texts, notes, and developed concept codes were studied adequately before comparing to SLR results and mapped with the interview Id of the interviewee into mapped tables. The mapping showed the commonality of data knowledge between literature and practitioners. The comparison can suggest which SA design decisions for ML systems are using in industry and what are missing in the academic literature. On the other hand, it also shows which SA design decisions for ML are in the literature and have not been discussed by the practitioners. The comparison of both data assists with the authentication of the mapped part of SLR outcomes.

4.4 Validity Threats:

Validity threats are listed in this section for both SLR and interviews. This section discusses three forms of validity threats. Internal validity is a problem that the researchers must

address internally, i.e., the cause and effect are determined within the analysis and are not clarified by other variables. Constructed validity relates to data and measurements: the quality and the measures. External validity risks are dependent on generalizable study results.

4.4.1 SLR Validity threats:

Interval Validity:
To find the related studies, I used a database search and the snowball method. Internal validity is jeopardized by the article selection method. This threat can be eliminated by reading papers and working with a supervisor to create and refine the search string. For this purpose, various database searches are conducted. After applying the inclusion-exclusion criteria, I deleted the unrelated articles. The backward and forward snowballing is used to find more specific studies. This procedure was carried out during the iteration. Following this process, I included more articles. For the evaluation of the obtained studies, quality criteria were created.

Construct Validity:
It is essential to create a search string to find specific studies. Ambiguity/Errors in search strings resulted in inaccurate results, posing a challenge to construct validity. I minimized this risk by identifying keywords in relevant posts. The supervisor also checked the search string to reduce bias in the collection of posts. In snowballing approach, I also remove the repeated studies.

4.4.2 Interviews Validity threats:

Interval Validity:
Internal authenticity threats for interviews are included as identifying practitioners for interviews, practitioner's knowledge of interview issues, and further analysis of data gathered. To minimize the risk of finding actual practitioners, professional ML system architecture development, ML system developer experience, personal industry contacts, and finding professionals through LinkedIn can be beneficial. There is a good chance that practitioners can miss the key point of the issue related to research questions. A simple interview questionnaire with examples was included to mitigate this threat.

External Validity:
To minimize the risk of not having generalized outcomes from interviews, diverse individuals from the industry with varying levels of expertise from various countries and in different positions were considered.

Construct Validity:
The construct validity of interviewing refers to achieving the data collection goal of the interview through a questionnaire. I brainstormed with the supervisor in preparing the questionnaire when considering study questions.

4.4.3 Limitations and Delimitations

Due to pandemics (Covid19), the survey interviews with the experts were conducted online. However, it was performed online due to pandemics. Another limitation is that the available literature might not define all main architecture decisions for ML systems. These might be based on relatively small sample size, challenges, and best practices are not general. Since July is considered the month of summer vacation in Europe, it might be possible that there were not enough experts for interviews. Another alternative solution that I considered is to conduct interviews in June and compare the interview results with the SLR results in July.

I did not distinguish the ML design decisions, such as designing ML systems and the ML model's pattern. I focused on SA designing decisions for ML systems and not ML model designs because this is not in the study's scope. The ML model design challenges and practices are out of the study's scope, but this can refer in future work. The wrong understanding of the interpretation, the expert's point of view might not be the same through an online meeting. The legitimacy of SLR results can be threatened by reliability and mostly internal validity [34].

Chapter 5

5
Results

5.1 SLR Data Analysis

I discussed the research findings from the 41 relevant selected papers in this chapter. Appendix A contains the primary study publications as well as the relevant reference numbers. The following procedure is used to choose the primary study. Research supports the SLR objective and if the studies describe the RQ1, RQ2, and RQ3 responses. One researcher split the papers evenly and used the inclusion-exclusion/search criteria stated above.

I started the study with 6494 publications, and after applying inclusion and exclusion criteria, I narrowed it down to 300 studies. I studied left papers abstract, introduction, methodologies, and conclusion to select or eliminate the papers. Total 41 papers were chosen through SLR and snowballing using four databases. The snowballing method was chosen because it is easier to locate papers than standard database searches, i.e., by utilizing forward and backward snowballing of the start set papers. The initial iteration of forward and backward snowballing, 15 more paper was selected. After applying the same procedure to read the paper, I chose seven related papers for study.

All the papers focused on the software architecture design, common challenges, best practices, and main decisions of ML systems. Following graph shows the percentage of selected papers in four databases:

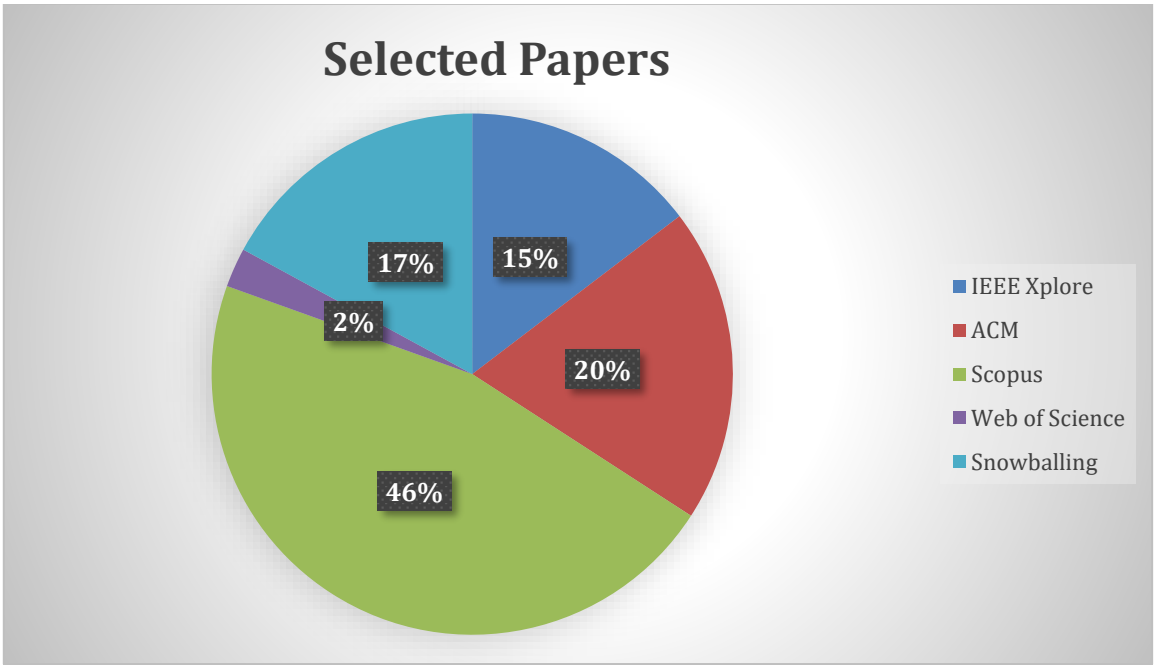

Figure 4 Selected Papers Percentage in Four Databases

5.1.1 SLR RQ1 Common Challenges Codes, Theme, and Description

A total of seventeen distinct topics were chosen by performing SLR on academic papers to identify the challenges of Software Architecture designing Machine Learning systems which refer to RQ1. The focus of the theme is "Challenges," which corresponds to the codes/topics.

The following table shows the common issue occurring while designing the software architect of the ML systems:

Id	Codes Common challenges/Issue	Theme	SLR Paper Id
SC1	Deductive verification and model checking are barely applicable to ML due to their inherently probabilistic and non-linear nature. *Description:* At the algorithmic level, various techniques increase qualitative qualities such as safety, robustness, or dependability. However, because ML is fundamentally probabilistic and non-linear, many traditional methods to ensuring system correctness, such as deductive verification and model checking, are only marginally relevant.	Challenges	S14
SC2	Data management (data pre-processing and preparation) in ML systems has significant data design issues. Data management design issues sometimes remain after the ML system is developed. For example, handling extensive data, including scientific imaging data using ML reflected complex relationships, it's challenging to manage a large amount of data and time-consuming to infer contextual information. The ML outcomes are mostly dependent upon the input data and its training. So large amount of managing data are a common challenge when designing ML systems.	Challenges	S15
SC3	Microservice architecture has multiple design challenges while developing an ML system using it. *Description:* Because the MSA (microservice architecture) application is separated into numerous services, locating the failure nodes is challenging. A service failure in the MSA (microservice architecture) may cause an avalanche effect, resulting in the failure of the entire system.	Challenges	S18

SC4	The following are the most common challenges, especially for ML-based safety-critical systems, and can apply to other ML systems too: • Model complexity and opacity. • Complete probabilistic output. • Sensitivity to variations in distribution. • Scenarios of limited testing. • Formal verification is either impossible or impractical. • Restrictions on Limited defect detection, prevention, and containment. • Strict reasoning regarding systemic flaws • Code reasoning is limited. • There is very limited heterogeneous redundancy. The introduction of complete probabilistic components with vast input and output spaces, on the other hand, creates common challenges. • To identify acceptable safety design patterns. • To justify the safety qualities attained by employing. • For picking the best patterns for the job at hand.	Challenges	S22
SC5	Neural network systems developed using ML have a design challenge to reusable their building block. *Description:* It is still challenging to use neural networks with ML as reusable building blocks with clear interfaces in productive systems.	Challenges	S26
SC6	The cyber-physical systems (CPSs) are the integration of computation, networking, and physical process. The CPSs developed using machine learning have design challenges in managing the system's continuous change and evolution of CPSs and operating environment. *Description:* The usage of multi-core architectures in CPSs is complex because its software interacts between cores, and its interaction design challenges are difficult to solve. The difficulty of design challenges occurs if the one software with dynamic behavior is	Challenges	S27

	allocated to the noncritical domain and further system interacts with critical applications in the static part.		
SC7	Mobile Robot developed with ML has major design challenges are to involve the integration of several different bodies of knowledge. One example of specialized bodies of knowledge is that computer vision does not properly employ a multitude of sensor technologies. *Description:* There are generally no acknowledged standards, guidelines to develop mobile robots using ML. The selection and design of an architecture for a mobile robot that satisfies the functional and quality attribute requirements is a big challenge. The designing of the localization and navigation part of the robot system is challenging because of less available knowledge of computer algorithms and probability theory.	Challenges	S28
SC8	The use of machine learning (ML) components in software systems introduce uncertainty. Uncertainty exists while evaluating the reliability of software architectures design. Prior information on the uncertainty of ML components employed at design time is regarded as incomplete and challenging. The usage of ML components in a software system can propagate and influence other system components.	Challenges	S30
SC9	SA still challenges the quantification of architectural expertise in ML systems. For Example, it's challenging to identify appropriate individuals in making suitable design decisions according to the requirements.	Challenges	S32
SC10	The input data can change over time, which can require updating in ML systems, especially the change in Model can add impact on system outcomes and its performance. *Description:* The dataset is input at the model training stage and emits a learned model that produces predictions. Dealing with the changed input data is challenging. However, the workflow becomes more challenging when ML systems need to be deployed in production. The comprise machine learning platform requires other components together with the learned model. The components offer automation to deal with a wide range of problems that can occur in production and model training.	Challenges	S34

	Creating this level of automation is difficult, and it becomes considerably more difficult when we consider the following complications: • Change in input data. • Creating a single machine learning platform for a variety of different learning tasks. • Continuous training and service. • Human-in-the-loop. • Reliability and scalability at the production level.		
SC11	Data dependencies design issues in ML systems are challenging, and configuration lines are usually more than traditional codes lines. *Description:* Hidden debts in the ML systems are harmful because it can affect the performance of the systems. Data dependencies in ML systems have a comparable potential for debt accumulation but may be more challenging to discover. In a mature system that is actively being developed, the number of configuration lines might considerably outnumber the number of lines of traditional code. The increment in configuration lines is challenging to handle.	Challenges	S38
SC12	Wrong selection of the model and not being aware of best practices can lead to poor performance of ML systems which is still a large challenge. *Description:* Model management consists of training, maintenance, deployment, monitoring, organization, and documentation of machine learning (ML) models. Incorrect model management decisions can result in poor ML system performance and high maintenance costs. Because both infrastructure and algorithm research is rapidly developing, there is a lack of knowledge of the problems and best practices for ML model maintenance. Authors classified the obstacles, but the key ones are: • Conceptual difficulties, such as determining what component of a model is. • Data management issues, such as problems regarding the abstractions used in ML pipelines, • Engineering problems include creating systems that use several languages and specialized hardware.	Challenges	S39

SC13	ML or Non-ML has different practices like a requirement, design, testing/quality, process, and management. *Description:* In a few areas, the design of ML systems and non-ML software systems and their everyday challenges differ. ML system architectures generally include data gathering, data cleansing, feature engineering, modeling, execution, and deployment. In contrast, non-ML software system architectural design is a more creative approach that implements different structural divisions of software components and provides behavioral descriptions. The distributed architecture style is commonly favored for ML systems due to the large volume of data. Complexity in architectural and intricate design is generally the result of a distributed architectural style. Secondly, ML systems have less emphasis on component low coupling than non-ML software systems. Even though various features of ML systems have distinct capabilities, development teams are closely linked. The performance of data modeling, for example, is dependent on data processing. Thirdly, detailed design for ML systems is more flexible than for non-ML software systems. It has been observed that data modeling might contain tens to hundreds of possible machine learning algorithms; as a result, the comprehensive design of ML systems would be time-consuming and iterative. Software developers often undertake a huge number of experiments in order to create an appropriate model.	Challenges	S4
SC14	The ML systems managing or versioning, model customization, and handling components are challenging. First, maintaining and versioning the data required for machine learning systems is far more complicated and demanding than other forms of software engineering. Secondly, Model customization and reusing the required abilities that are not commonly found in software teams. Third, ML/AI components are more difficult to manage as separate modules than typical software components because models can get "entangled" in complicated ways and exhibit non-monotonic error behavior.	Challenges	S8
SC15	Deep Learning systems that use Machine learning software architecture have various design challenges. A deep learning system is	Challenges	S16

	extremely demanding for a computer system's hardware. It costs a lot of time, severely limiting their practical application, such as embedded and real-time systems. Modern CNN designs with ML systems are highly resource-demanding, which limits their practical usefulness. Furthermore, big resource consumption may be solved utilizing cloud computing and large data processing centers. However, this is not appropriate for all circumstances, such as when a crewless vehicle loses contact with a distant server, resulting in an accident.		
SC16	The ML design smells (may indicate an underlying problem in a component/system) are difficult to find. The design smells can produce in several ways; for example, using multiple languages in the ML system often increases the cost of effective testing and makes it more difficult to transfer ownership to other team members. Another design smell is maintaining the prototyping environment is costly, and small scale rarely reflects reality at full scale.	Challenges	S38, S39
SC17	The machine learning systems with three-layer architectural patterns consist of the presentation, logic, and data layers. The common ML systems with three-layer architecture complicated the troubleshooting method because of their tightly coupled functions, for example, business logic code from design and inference engines derived from data. On the other side, three-layer architecture to handle the change in input is difficult. The instability in input data jeopardizes operational stability, resulting in model staleness and missing values. When model staleness occurs, inference engine performance suffers as a result of changes in input data trends.	Challenges	S6

Table 2 SLR Results of RQ1

5.1.2 SLR RQ2 Best Practices Codes, Theme, and Description

A total of twenty-five distinct topics were extracted to identify the best practices in architecting and designing machine learning systems that refer to RQ2. The focus of the theme is “best practice,” which corresponds to the codes/topics.

The following table shows the best practices in architecting and designing machine learning systems:

Id	Codes Best practices	Theme	SLR Paper Id
SP1	The best design designing practices in ML autonomous systems like advanced robots and self-driving [vehicles] to add certifiability, safety, time predictability, and security. The following explains all four aspects: Certifiability: All safety-critical software components must be written in accordance with strict coding standards and certified by appropriate certification organizations. Safety and Fault-Tolerance: The system can avoid catastrophic consequences for the user(s) and the environment by implementing proper mechanisms for tolerating faults and failures that may occur in complex software routines. Time Predictability: ML autonomous systems must respond to environmental events within predefined time boundaries, which are computed at design time based on a set of performance requirements. Security: The software must be designed to defend the system from cyber-attacks that could target vulnerable sections of the code in order to alter the software and gain control over the system.	Best practice	S16
SP2	The best design practice to remove distortion on ML scientific imaging databases is to use in-memory disturbed learning architecture. The in-memory distributed learning architecture can achieve by applying sophisticated learning and optimization techniques on scientific imaging datasets.	Best practice	S17
SP3	Microservice architecture's (MSA's) security concerns are receiving much attention as its popularity grows. The best design practice for mining causality using MSA is to divide into invocation chain anomaly analysis using robust principal component analysis and a single indicator anomaly identification algorithm. The principal component analysis method is known to assists in reducing the dimensionality of datasets and extend interpretability.	Best practice	S18
SP4	ML-based digital forensics systems are part of forensic science that deals with recovering and investigating the information contained in digital devices. The best practice for developing digital	Best practice	S20

	forensics systems comprises four phases: seizure, acquisition, analysis, and reporting. The following is the explanation of four phases: Seizure: First, determine which devices may contain useful information for the ongoing investigation and then seize. Acquisition/imaging: Create an exact duplication of the data, and analysis of the data is recommended to perform on the duplicated data. Analysis: The image of the device can be analyzed to identify your results using ML techniques. Reporting: After data analysis, the results can be described in an official report.		
SP5	In safety-critical systems using ML, the best practice is that the designers must focus that the design practices don't let a system in reaching a hazardous state (caused more failure). *Description:* The best practices of software architecture designers must guarantee that software employed in safety-critical systems developed using ML does not cause or contribute to the system reaching a hazardous condition. When a defect is discovered, the safe state must reach quickly - for example, by turning off components of the system. In other cases, the system should continue to function securely despite any failures or allowing some graceful deterioration - until a viable, safe state is reached.	Best practice	S22
SP6	The Federated learning simulation framework (FLSim) algorithm is the best design practice for simulators, which helps developers create simulators in different scenarios efficiently. *Description:* The FLSim is a federated learning simulation framework, and it's an extensible and reusable simulation framework for federated learning. FLSim commonly assists deep learning, machine learning frameworks, for example, PyTorch and TensorFlow. It's recommended to use FLSim in creating simulators that have any of the above-mentioned frameworks.	Best practice	S24

SP7	The advantages of using ML cloud technologies increase the quality of cutting tools states recognition in the industry.	Best practice	S25
SP8	The best design practices of the Artificial neural network in ML systems are the appropriate selection of modeling languages. Three concerns are network architecture, network training, and dataset model are recommended to consider during the selection of modeling language. The following are the explanation of three concerns: Network Architecture: The first thing that comes to mind when designing a neural network using ML is its actual architecture, which consists of neurons, mostly organized as layers, and connections between the neurons that define the data flow. Network Training: Even the best network architecture for a specific task is useless if it is not properly trained. The developer can modify the training procedure without changing the architecture, or the developer can combine existing architectures and training models without changing the models at all. Dataset Model: Finally, the compiler needs to know where to look for training data and how to load it to train a network. Furthermore, the dataset must be divided into training and test data; developers recommended only using parts of the data or skipping training if the dataset has already been learned.	Best practice	S26
SP9	The Siemen's four-view architecture designing approach is based on best architecture design practices for mobile robotics development using ML components. This approach assists in reducing the complexity of architecture design activities. The Siemen's four views architecture approach are conceptual view, module view, execution architecture view, and code architecture view. These four views increase the simplicity in the development of the mobile robotics system developed using ML.	Best practice	S28
SP10	It is best to design practice to explicitly model the intrinsic uncertainty particular to ML components and assess how it propagates and impacts other elements in the system at the designing stage.	Best practice	S30

SP11	In the best designing process, the architects and developers can favor a naturalistic approach in selecting appropriate designs.	Best practice	S32
SP12	The best practice is to use TensorFlow-based learner implementation with support for continuous training and serving with production-level dependability. Id 16 identifies that the primary strategy of TFX is to efficiently orchestrate and give users a unified configuration of recurrence components (data analysis and transformation, data validation, model training, model assessment, validation, and infrastructure service).	Best practice	S34
SP13	The best designing practice that can assist in designing ML systems is documentation that improves the ML System's reusability. *Description:* Reuse, extension, and usage of undocumented software frameworks are difficult to envision. So, the best practice is to properly describe documentation and code, which can increase reproductivity and shareability.	Best practice	S35
SP14	Using a single container, single-mode patterns, and multiple-node patterns is the best design practice in reusing components and dividing implementation among teams. Following are the explanation of the above design patterns: 1) Single-container patterns (Single- container pattern provides a natural boundary for defining an interface.) 2) Single-node patterns (Single-node patterns comprise symbiotic containers co-scheduled on a single host machine). 3) Multi-node patterns (multi-node patterns help to combine multiple containers into a single pod.) The advantages are as follows: • It’s simple to distribute implementation across various teams and reuse components in new situations. • Distributed systems include unique features, such as the ability to upgrade components separately and the ability to write in several languages. • For the system to decline gracefully as a whole.	Best practice	S36

SP15	ML systems are typically composed of several workflow pipelines. Due to time constraints, a pipeline is often quickly constructed to meet a specific requirement, which can be caused to make a Pipeline jungle. Pipelines may include various glue codes written in various languages as well as special languages for managing pipelines. The best design practice is to avoid Pipeline jungles by taking a step back, looking at systems as a whole, and taking data collection and feature extraction seriously.	Best practice	S38
SP16	The best practice to find the ML design smells is to focus on three types of ML design smells: Plain-Old-Data, Multiple-Language, and Prototype smells. The following is the description of three types of design smells: • The smell of Plain-Old-Data Type (Complex information used and produced by ML systems is all too often encoded with plain data types such as raw floats and integers.) • Smells in a Variety of Languages (Using multiple languages often increases the cost of effective testing and makes transferring ownership to other team members more difficult.) • The smell of a Prototype (Maintaining a prototyping environment comes at a cost, and there is a significant risk that time limits will lead to using a prototyping system as a production solution. Furthermore, results obtained at a small scale rarely reflect reality at full scale.)	Best practice	S38, S39
SP17	The best design practice to minimize the technical debt in ML systems is to focus on debt for Data testing, the debt of reproducibility, process management, and debt to culture. The following is the explanation: • Debt for Data Testing (If input data requires to replaces the code in ML systems, then code should be tested once again. Commonly, some input data testing is critical to a well-functioning system) • The debt of Reproducibility (It is difficult for team development to be able to repeat experiments and obtain similar results) • Debt in Process Management (Process Management debt significant issues are to update the configuration of similar	Best practice	S38

	models, it’s an issue to visualize and detect blockage in the flow of data) • Debt to Culture (It is essential to encourage team cultures that value feature deletion, complexity reduction, reproducibility, stability, and monitoring improvements in the same way that accuracy improvements are valued)		
SP18	The large-scale forecasting problems and time series prediction systems developed using ML gained attention. The best design practice to develop a similar system is to train a single model per time series and retrain the models every time a new forecast needs to be created. Some classical forecasting techniques can be used to develop a similar system, such as Auto-Regressive Integrated Moving Average models, exponential smoothing methods, and state-space formulation.	Best practice	S39
SP19	The change in input data usually requires an update or extension of the ML system's functionality to accommodate the new changes. The visualization of data assists the development team in understanding the previously trained model data as well as new data. The best design practice is to use TensorFlow. Because of its graphical approach, TensorFlow can provide a better way of visualizing data. It is difficult to effectively resolve the neural network in the ML system. However, TensorFlow assists the development team in debugging the nodes with the help of Tensor-Board, which reduces the effort of visiting the entire code.	Best practice	S40
SP20	The environment influences the ML-based mobile robot's recognition accuracy. For example, the recognition rate is higher in more light. The best design practice to increase the efficiency of recognition is that when the intensity of the light changes, the noise cleaning algorithm's parameters are recommended to be adjusted to account for the change. Simultaneously, to complete image enhancement, the composition model of related algorithms may need to be reconstructed. Furthermore, all pattern recognition algorithms and the composition model builder should be capable of checking the information of related sensors when necessary.	Best practice	S1

SP21	ML and Non-ML system best practices are different as both follow the different design, implementation, deployment, and maintenance techniques. The best practice while designs an ML system is to distinguished ML and Non/ML systems and take both designing and development techniques separately.	Best practice	S4
SP22	The troubleshooting process in ML systems is difficult because of tightly coupled functions, for example, inference engine derived from data and business logic code from design. The best design practice is to separate business logic and machine learning components. The usage of three-layer architecture can assist the development team in breakdown the failure into a business logic part and an ML-specific component part; it can also allow to roll back the inference engine independently of the business logic when the inference engine encounters some issues. *Description:* The usage of three-layer architecture for machine learning systems separates business logic and machine learning components to tackle the troubleshooting challenges. It can enable operators to split out failures into a business logic portion and an ML-specific part. The inference engine may roll back independently of their business logic when there are specific difficulties with the inference engine.	Best practice	S6
SP23	The best design practice is to train a partial model using existing general datasets (e.g., ImageNet for object identification) and then combine it with additional specialized data using transfer learning for better performance of the object identification system.	Best practice	S8
SP24	A significant design concern best practice is to divide it into sub-concerns, which are subsequently mapped into design practices within each subsystem. *Description:* The ML design techniques as significant design issues deconstructed into sub-concerns, subsequently mapped into each subsystem's design decisions.	Best practice	S11

SP25	Static analysability, Monitor Analysability, A-Posteriori Analysability, and Non-Analysability are the best practice example of preliminary classifications. The usage of preliminary classification practice can provide confidence to their design process for ML systems.	Best practice	S14

Table 3 SLR Results of RQ2

5.1.3 SLR RQ3 Main Software Architecture Design Decisions Codes, Theme, and Description

The eighteen topics/codes were identified, representing the main software architecture design decisions of Machine Learning systems that target our RQ3. The theme's focus is "Main Software Architecture Design Decisions Guidelines," corresponding to the codes/topics.

Following table shows the main design decisions guidelines while architect the software architecture design of the machine learning system:

Id	Codes Main Decisions Guidelines	Theme	SLR Paper Id
SM1	The use of Infrastructure as Code (IaC) in ML design decisions helps reduce the cost, time, risk of IT infrastructure. *Description:* The Infrastructure as Code is known as the method of managing and providing computer data centers through machine-readable specification files, rather than actual hardware setup or interactive configuration tools, which makes it better than other methods. The main design decision of using the IaC in those ML systems associated with Databases, servers, and other IT infrastructure then helps manage the operations using the same structures and rules used for code development. Security concerns arise when systems deal with databases and servers, but the usage of IaC can enhance the security of the systems.	Main Software Architecture Design Decisions Guidelines	S15
SM2	The main design decision in ML-based safety-critical components is that all safety-critical software components must be written according to strict coding standards and certified by appropriate certification organizations.	Main Software Architecture Design Decisions Guidelines	S16

	The other alternative is to perform validation testing, but it will be more secure to follow coding standards and approve certification together with testing because safety-critical systems failure can cause significant damage to property, environment, or loss of life.		
SM3	The ML-based large-scale learning system requires more focus on facilitating data analytics. The main design decision is to add data visualization techniques in the designing process, which can express the relationship between data and computing tasks. The other alternative is to perform proper data cleaning or labeling, but there is a chance to skip a chunk of data.	Main Software Architecture Design Decisions Guidelines	S17
SM4	The major benefits of using discrete services are reducing system coupling and providing more flexible service support. The main design decision to decompose a big service into discrete services is to use microservice architecture in ML systems. The other alternatives to use Keras and Pytorch, but MSA can provide better results	Main Software Architecture Design Decisions Guidelines	S18
SM5	ML-based digital forensics systems are part of forensic science that deals with recovering and investigating the information contained in digital devices. The Digital Forensics system using ML has four phases: seizure, acquisition, analysis, and reporting. All four phases are explained in table 3, Id SP4. The aim of the main design decisions is to focus on four phases of digital forensics systems in the design-making process. The division of forensics systems into four phases can provide more designing simplicity than applying normal development stages.	Main Software Architecture Design Decisions Guidelines	S20
SM6	The ML systems which are developed through space data are usually contained a large amount of data. The main design decision is to use concurrent engineering methods and model-based system engineering while analyzing the space data and space mission system design. The other alternatives are slow with dealing with spatial data. For example, the Sequential engineering method runs in a	Main Software Architecture Design Decisions Guidelines	S21

	linear format, and each step is taken before moving to another. On the other hand, the concurrent engineering method and model-based system engineering can deal with big space data and divide it into different stages. The different stages can run simultaneously, which assist in low development time, cost and enhance productivity.		
SM7	When existing patterns are not feasible for the ML-based safety-critical systems, the main design decision for developing such a system is to create new patterns. The main design decision for the ML-based safety-critical systems is to include the evaluation process of architectural safety methods before moving to the next development stage because the failure of safety methods can damage the environment or property.	Main Software Architecture Design Decisions Guidelines	S22
SM8	The ML-based mobile computing system's main design decision is to use federation learning. This can overcome the challenges by enabling continuous learning on end-user devices and ensuring data is not lost in end-user devices. Federation learning can also help to the preservation of data privacy in the ML system. Federation learning is better than other alternatives. For example, the distributed learning is usually trained on a complete data set, but the end-user device doesn't have complete dataset access, which is an issue.	Main Software Architecture Design Decisions Guidelines	S24
SM9	The usage of cloud technologies is performance-wise faster in cutting tools state recognition systems. It's better than other alternatives, for example, diagnostic feature selection using combinatorial analysis, but this approach requires more computing power. The main design decision to use ML cloud technologies in SA design of cutting tools states recognition systems because of high-performance computing.	Main Software Architecture Design Decisions Guidelines	S25
SM10	The major modeling design architecture decisions concern modeling and training neural processing in ML systems to help achieve better performance systems.	Main Software Architecture Design Decisions Guidelines	S26

	Description: The differentiated three key modeling issues result from architecture, training, and data. Instead of dealing with low-level constructs, a robotics specialist prefers a language that accurately represents their area. For example, deep learning technologies have been more accessible by expressing layered structures as YAML or prooftext descriptions or offering high-level Python interfaces like Keras and Lasagne.		
SM11	There are many proposed and practiced architectures for ML-based mobile robots, e.g., Layered Architecture, Implicit Invocation, Blackboard Architecture, Control Loop Architecture. The Siemens 4 view approach is appropriate in the main design decision when developing the robot navigation component because of its recurring activities. This approach also assists better in determine the movement/action plan for the avoidance of obstacles in the path of robot motion.	Main Software Architecture Design Decisions Guidelines	S28
SM12	The article discussed the design methods to consume fewer resources, including hardware and time cost for CNN and other ML computation operations. *Description:* Reduce the number of resources used, both in terms of hardware and time, for Convolutional neural networks (CNN) and other ML computing operations are described in this article.	Main Software Architecture Design Decisions Guidelines	S41
SM13	Model validation is essential because it is difficult to predict whether a learning algorithm will behave reasonably on new data. So, in the main design decision, model validation must consider and check the validation results before pushing into the production environment. There are several ways for model validation, but the best is to combined model validation with data validation to detect corrupted training.	Main Software Architecture Design Decisions Guidelines	S34
SM14	The components-based machine learning distributed system is a multi-node machine learning system that can	Main Software Architecture Design	S36

	increase efficiency and improve the system's performance by handling large-scale input data and ML components. The components grow in machine learning systems, and alternative approaches like principal component analysis can solve the complexity of components to divide components into the new component. The new addition of components can increase complexity, and it's time-consuming to handle new components. So, the main design decision of using the multi-node approach allows disturbed ML systems components to be upgraded independently, allowing the development team to handle the component if it grows. These patterns also allow writing components in a mixture of language, which is easy to update later.	Decisions Guidelines	
SM15	The configuration of a machine learning system is hard to modify; the configuration mistakes can be costly, waste of computing, leading to serious time loss, and might be a production issue. The main design decision to eliminate the configuration issues in ML is to develop all models separately, which will help to visualize the difference in configuration. The other approaches can help identify the configuration issues but are not efficient; for example, the debugging model training approach can help detect common errors during model training, but it's time-consuming and costly to eliminate the configuration issues. On the other hand, develop all models separately will also help to detect unused or redundant models in the ML systems.	Main Software Architecture Design Decisions Guidelines	S38
SM16	In the main design decision, the usage of TensorFlow for large-scale ML systems is beneficial. The main design decision is to use TensorFlow when the development team is looking for a higher level of performance that's also powerful and easy to scale, for example, voice/sound recognition, image recognition, and video detection systems. The other approaches like Keras and PyTorch are better for small systems because these are slower and lower in performance and speed.	Main Software Architecture Design Decisions Guidelines	S40

SM17	The environment changes can be challenging for the performance of ML-based mobile robot software. The other alternatives approach like more feature extraction, color constancy, and threshold area elimination method approaches are not enough to increase the efficiency. The main decision can be to perform the change in parameters of algorithm and model composition according to the environmental scenarios. For example, the noise cleaning algorithm should update its parameters to adapt to the environment changes when light intensity changes. Similarly, while changing the parameters, the composition of related model algorithms also needs to be reconstructed, which will help in image enhancement.	Main Software Architecture Design Decisions Guidelines	S1
SM18	The selection of models is the most important major design decision in almost all ML systems. An appropriate model selection is a better starting point when designing a better ML system because alternative trending models to use sometimes can't help to fulfill the system requirements. So, the models must choose based on the type of ML system, such as a batch system or a real system, and model to provide the best performing with useful outcomes. For example, the use of EfficientNet is better than the ResNet model approach in image classification because of scaling the dimensions of the image by a fixed number of layers that can provide better performance in real-time ML systems.	Main Software Architecture Design Decisions Guidelines	S8

Table 4 SLR Results of RQ3

5.2 Interview Data Analysis

A total of twelve interviews were performed with individuals with a mix of academics and industrial ML professionals. All mixed practitioners were asked the same interview questions, with additional follow-up questions to understand the answers better. The selection of interviewees and coordination is discussed in chapter 4 under the survey. Before the interview, the practitioners were informed that no personal information would be revealed or published and that the gathered evidence would be deleted after the research was done. The interview data were analyzed using a five-phase theme analysis technique [35].

I recorded all interviews after getting permission from the interviewee. I used the online software name Otter.ai to extract text from the video for further analysis. Phase one involves listening or watching the gathered material and taking notes while listening or watching interview transcripts. It is also necessary to make meaning of the participant's experience while taking notes. Phase two included developing codes for data that may be relevant to the study questions. Codes are created using a descriptive and interpretive approach to data. Every piece of data that is relevant to the study topics has been coded. Continue reading data after the first code has been recorded until the next possible required data are found that can be applied to the existing code or a new code. Phase three includes converting codes into themes that capture essential information about the research problem. The data from phase two was analyzed to see where there were similarities and commonalities across the codes. Phase four entails a study of prospective topics, often known as theme mapping. Each subject was re-evaluated to ensure in light of the facts. The fifth phase is identifying and naming themes. It was made sure that each topic was unique and that it addressed the study question directly.

5.2.1 Interview Statistical Representation

Table 5 shows the statistical representation of all 12 interviews. Every interview has a practitioner id of 'I' with an incremental suffix. Based on the availability of practitioners in different time zones, all online interviews are conducted utilizing the Zoom technology. Each interview took around 30-40 minutes, but some interviews took more time because of more discussion related to research questions.

Interviewee Person Id	**Role**	**ML Exp**	**Non-ML Exp**	**Total Exp**	**Place**	**Academic/Industry for ML**	**Company/University**
P1	ML Engineer	1 Year 8 months	3 Years 2 months	5 Years	Singapore	Industry	Hidden (Confidential)
P2	ML Engineer	1 Year	11 Years	12 Years	Egypt	Industry	Upland Software
P3	Researcher in ML	3 Years 6 months	3 Years	6 Years 6 months	Sweden	Academic (Previously worked in Industry)	Chalmers University
P4	Researcher in ML	26 Years	0 Years	26 Years	Italy	Academic (Previously worked in Industry)	University of L'Aquila
P5	ML Engineer	4 Years 5 months	5 Years 7 months	10 Years	India	Industry	Quantiphi

P6	ML Architecture Engineer	23 Years	0 Years	23 Years	Australia	Industry	PlayGround XYZ
P7	Researcher in ML	7 Years	0 Years	7 Years	Sweden	Academic	Chalmers University (director of Chalmers AI Research Centre)
P8	ML Architecture Engineer & Researcher	27 Years	0 Years	27 Years	USA	Industry	Cubic Corporation
P9	Researcher in ML	1 Year	2 Years 5 months	3 Years 5 Months	Pakistan	Industry	Cresta
P10	ML and Data Science Engineer	4 Years	12 Years	16 Years	Dubai	Industry	Inception Institute of Artificial Intelligence
P11	Senior Manager ML Architecture Engineer	15 Years	0 Years	15 Years	Australia	Industry	Macquarie Group
P12	ML Architecture Engineer	4 Years	0 Years	4 Years	Pakistan	Industry	Visionet Systems Inc.

Table 5 Interview Statistic

The interview participants experience is presented in the following bar graph:

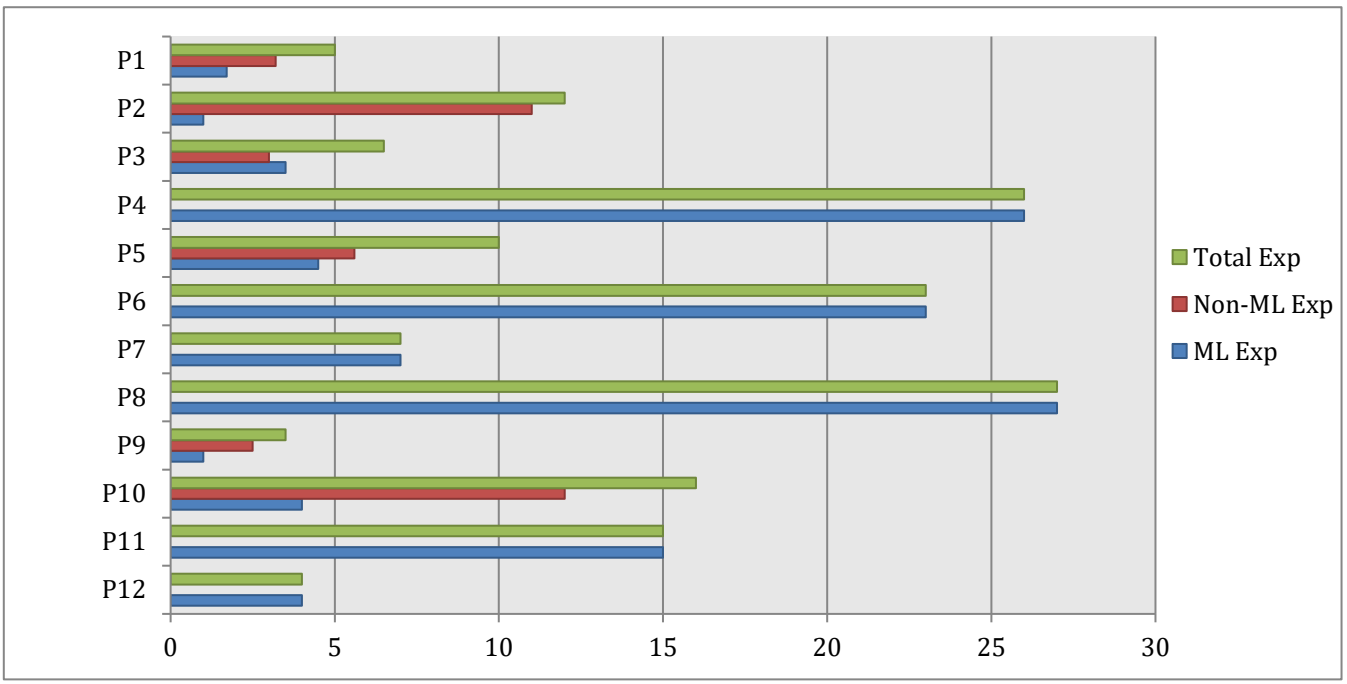

Figure 5 Interview participants Experience

The interview participants average experience is presented in the following bar graph:

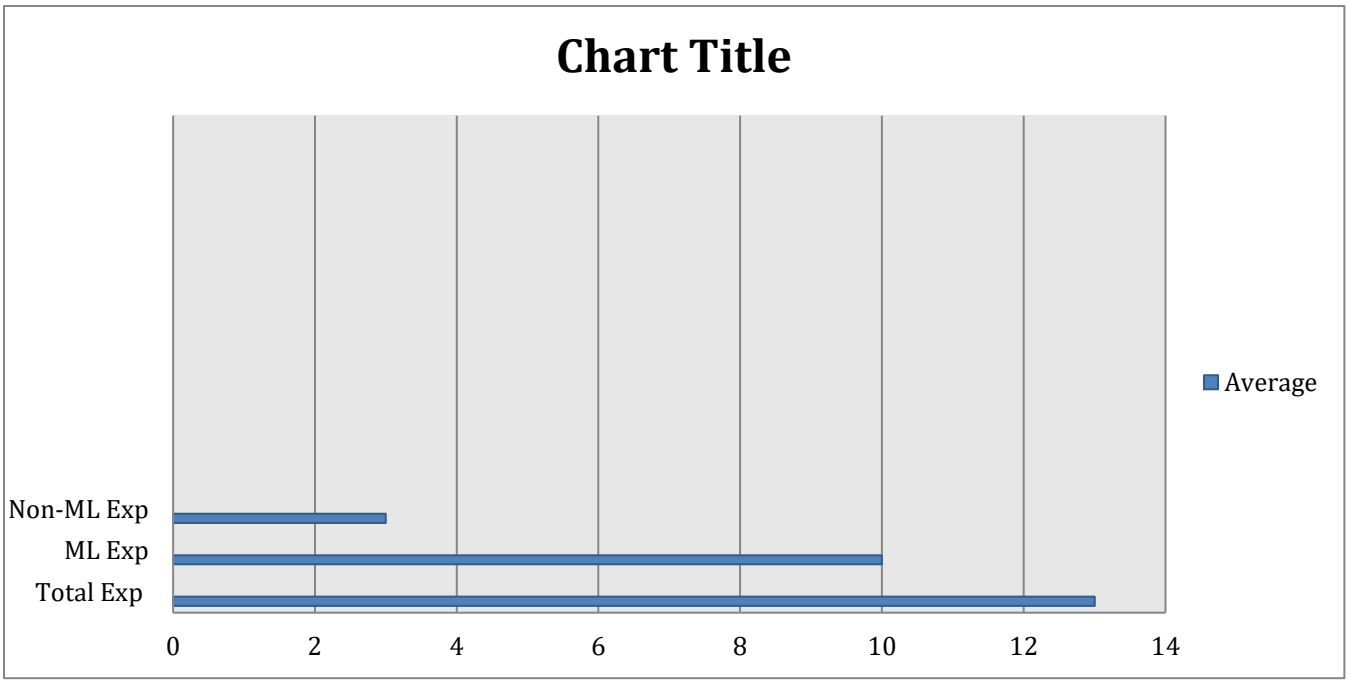

Figure 6 Average Interviewee Experience

5.2.2 Interview RQ1 Common Challenges Codes and Themes

Interviewee Person Id	Codes Common challenges/Issue	Theme
P1	Data are increasing, so does the model size is increasing. Common Challenges: • Observability of the data. • Expansibility (why the system makes a prediction). • The usage of databases is confusing.	Challenges
P2	• Cleaning the data and make sure data does not break the algorithm or be biased. • The input data visualization is difficult to perform, bioinformatic information is not common, and current techniques are not applied.	Challenges
P3	ML's common challenge/issue is the privacy of the data.	Challenges
P4	ML developing team needs to understand the data and also to understand the ML model because sometimes development team has lack of knowledge.	Challenges
P5	• Data accuracy and completion of data for training of ML systems. • Cloud function (lambda function) is good, but if high data are coming, it doesn't work; then the development team needs to work on other techniques. From a security perspective, the development team needs to choose other techniques; this is a major challenge.	Challenges
P6	Batching scoring is sometimes used to save to database, which is a challenge to save and take the data.	Challenges
P7	• Establish ML infrastructure – how to manage data (store, access, update) • Establish run-time architecture (distributed execution of ML model) and establish a relation to the ML infrastructure.	Challenges
P8	Data preparation is difficult; make statistics of the data preparation is difficult.	Challenges

P9	How to update the model in production, like to add the new data and to remain the same structure, are challenging to know.	Challenges
P10	• Most architects don't follow architecture services. The development team doesn't follow continuous integration, which becomes complex at the time of system completion. • ML common challenges in designing less available documentation, and the ML quality assurance (QA) area is lacking.	Challenges
P11	ML and Non-ML processes are different, and their common challenges are different, and the decision must not be followed from the non-ML system in designing and development.	Challenges
P12	Design challenges are not being able to scale the model and server is busy and not replies to your answer; make the queue system as your request can be lost. Model request vanishing can be the problem.	Challenges

Table 6 Interview Results of RQ1

5.2.2.1 Interview Analysis of the Software Architecture Design Common Challenges in Machine Learning

Total 12 interviewee practitioners offered their point of view related to common decision challenges in machine learning systems. The most frequent response is data management, lack of understanding of the business when making the design decision of ML systems. Irrelevant model and architecture selection is the most known issue in machine learning systems. ML systems are different than the software engineering system as it depends upon the data input and getting useful outcomes from it. Cleaning or labeling the data are the most typical problem, and it necessitates more potential than going forward.

Most practitioners expressed concerns about model upgrading. One known issue is if data are increasing and so does the mode size is increase; secondly, if data are corrupted, inaccurate, and new data are added, then the model gets affected, and sometimes it's difficult to remain the same structure.

Two practitioners identified a weak design decision, one of the most prominent challenges, recommending that a researcher/architect make ML system design decisions with the development team's help.

5.2.3 Interview RQ2 Best Practices Codes and Themes

Interviewee Person Id	Codes Best Practices	Theme
P1	The latest model is the trend to follow, but the most focus should be given to analyzing the data. Because the data are not correct, the latest model will not help.	Best practice
P2	Build the grids for developing natural language techniques. Grid is a data preparation technique that treats data transforms as another hyperparameter of the modeling pipeline. The grid searching data preparation technique suits more for natural processing language.	Best practice
P3	• The usage of microservice architecture in designing ML systems is beneficial because the use of microservice architecture in the design of machine learning systems is simple to maintain as it assists in splitting the system into a set of components. MSA enables the development team to concentrate on building business functionality rather than writing glue code. • Unified architecture, classified approach, small package, and try not to make too many architectures decision before development, move the architecture decision close to function developer are the best design practices.	Best practice
P4	The knowledge of the technical depth and understanding the process of creation and training of the model according to systems requirements are the best design practice.	Best practice
P5	I recommend that ML-based natural language systems use the Microservice architecture and not the blackboard because microservice can provide better results.	Best practice
P6	Define the standardization of the training and testing process makes it easy to validate the models.	Best practice
P7	• Most companies use DevOps and scrum for software development and have different variants of ML workflow. • Make ML designing planning according to the product type, the organization, and business goals.	Best practice

	• It is important to ensure proper ML infrastructure and the training & deployment process, which might impact the SA. The application domain depends on it; for example, Federated Learning requires distributed system/software architecture. Here we need to design a distributed, dynamic system. Experience in architecting distributed systems and cloud computing would be useful.	
P8	• Pipelines architecture acts plug and plays for replacing the component its modular architecture, which is easy to work with. • A pipeline architecture with TensorFlow is better to use in ML systems that can enhance the security of the ML system.	Best practice
P9	For general, ML systems use test-driven development, Object-Oriented way of development/techniques, and Jupiter Notebook as best practices.	Best practice
P10	• For designing ML systems, software engineers and research need collaboration for planning. • Object recognition and image processing systems use client-server architecture, low coupling, and high cohesion techniques to place data behind the firewall. • Based on the requirement, we must have comparative analysis, data-driven approach, or test-driven approach, or data at the initial level. Use Client-Server architecture and autonomous, so we don't break any pipeline • ML designing requires most collaboration from developers who worked in the existing system; according to my experience, researchers generally do not provide the right solution, but practical experience helps design ML systems.	Best practice
P11	• While building the framework and software architecture design model, try to avoid new hiring to work on it as in some cases, it can lead to change/update in data architecture. • When building on ML framework, we have to confirm the architecture remains the same or choose which technology changes it doesn’t affect the framework. • Create a small system prototype in 1-3 sprints, and when the business is happy, make a software design and proper development and then deployment.	Best practice

P12	• If ML systems use microservice architecture (MSA), then modifying your model and releasing the new model is very useful because of low cohesion and low crash. MSA helps to ML system perform long prediction and short prediction. • Look at the business and see it's real-time or not real-time before deciding to design and develop ML systems.	Best practice

Table 7Interview Results of RQ2

5.2.3.1 Interview Analysis of the Software Architecture Design Best Practices in Machine Learning

Most practitioners recommended optimal practices in ML include following techniques of data cleaning and proper labeling; for example, if you are using the latest model and still, it will not help you because the data are not correct. Some of the most significant components of ML are the Selection of architecture, pipelines, tools, optimization of model techniques, training and deployment process, and its impact on the software architecture of the ML systems.

5.2.4 Interview RQ3 Main Software Architecture Design Decisions Codes and Themes

Interviewee Person Id	**Codes Main Decision Guidelines**	**Theme**
P1	The main design decision while designing the software architecture of the ML system is to make the consideration like who is the end-user and system fulfill the business needs. So, first, make a prototype of a system and feedback from stakeholders is important because another way of start development and ask for feedback from stakeholders. If stakeholders are not satisfied, then it’s costly and a waste of time.	Main Software Architecture Design Decisions Guidelines
P2	The major design decision is to always build the grids for ML-based natural language systems, which enhances its performance; if you couldn’t find a solution online, it is better to create your packages for ML systems. The alternative is to modify the packages	Main Software Architecture Design Decisions Guidelines

	according to your system needs, but this is risky because it's time-consuming in case of failure.	
P3	The architects are required to make design decisions with the collaboration of the function developer. The main reason to consider it as the main decision is that developers have practical experience, and their expertise can help make better design decisions.	Main Software Architecture Design Decisions Guidelines
P4	The development team must consider the evolution of the ML system as a major design decision. The evolution requires input data change, and then the development team needs to change the ML system's software architecture decision accordingly. The other alternative solution is to retrain the model with new input data and deploy it in production, but the risk remains the same, and whole system outcomes might not fulfill the business needs.	Main Software Architecture Design Decisions Guidelines
P5	The ML development teams need to make a general model, and then their main design decision focus must retain it because the newly hired team member can change the model, which can impact the ML system performance.	Main Software Architecture Design Decisions Guidelines
P6	The most important major design decision in almost all ML systems is the selection of a model. The model selection must depend upon the ML systems domain type like batch system or real system. The appropriate model selection is a better start to design a better ML system. The appropriate model selection is a better design decision to use because it can give a good designing start of the ML system.	Main Software Architecture Design Decisions Guidelines
P7	The main decisions are related: how to manage and access data, how to define the ML development architecture, how to define the runtime architecture with heterogeneous computing units (CPU + GPU), how to enable a continuous ML, etc.	Main Software Architecture Design Decisions Guidelines

P8	When the ML development team uses the TensorFlow Extended architecture in their software architecture design decision, the main design is to use pipelines to enhance the ML system's security and efficiency. If we don't use the pipelines, then data privacy flaws can grow.	Main Software Architecture Design Decisions Guidelines
P9	The usage of microservice architecture (MSA) is appropriate in developing ML-based natural language processing systems because the other blackboard approach is not better in data cleaning. The main design reason to use this MSA architecture is that it can assist in performing a better data cleaning which helps the accurate parsing of the document.	Main Software Architecture Design Decisions Guidelines
P10	The main design decision is to use the client-server architecture for object recognition and image processing systems because it's more appropriate than other architecture. The client-server architecture provides low coupling and high cohesion. This architecture helps to enhance the system's security; for example, it helps to place the data behind the firewall.	Main Software Architecture Design Decisions Guidelines
P11	The amount of input data can impact the performance of the ML system. So, the main design decision is to select the architecture according to the input data to train. When input data are frequently changing and the system is large, one main design decision is not to use microservice architecture as it's not appropriate. The other alternative is to use the Keras, but it's slow in performance and speed but works fine infrequent data changing.	Main Software Architecture Design Decisions Guidelines
P12	The main design decision in the ML system is the training of the pipelines must be in separated branches and not involve in training the model because it's easier to integrate into large codebases and easier to share with team members. The other alternative is pipelines training can be in the same branch, which can reduce time cost, but it can cause in the making of pipeline jungles.	Main Software Architecture Design Decisions Guidelines

Table 8 Interview Results of RQ3

5.2.4.1 Interview Analysis of the Main Software Architecture Design Decision in Machine Learning

According to the interviewee practitioners, the main decision is based on the business requirements or end-users and the type of ML system, such as static or adaptive systems. It's frequently recommended that main designing decisions depend upon handling and accessing data, creating a machine learning development architecture, defining a run-time architecture with heterogeneous computing units (CPU + GPU), and allowing continuous machine learning.

Chapter 6

6
Discussion

6.1 Discussion

Following are the SLR and interview combined results:

6.1.1 SLR-Interview-RQ1 Common Challenges Mapping

Id	Common Challenges SLR Id	Interviewee Person Id	Source
SIC1	SC2	P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11	SLR, Interview
SIC2	SC3	P10	SLR, Interview
SIC3	SC8	P4, P10	SLR, Interview
SIC4	SC9	P10, P11	SLR, Interview
SIC5	SC10	P6, P9, P12	SLR, Interview
SIC6	SC12	P3, P4, P5, P9, P10	SLR, Interview
SIC7	SC13	P11	SLR, Interview

Table 9 SLR plus Interview Results of RQ1

6.1.1.1 Answer of RQ1

What are the common software architecture design challenges in machine learning systems?

This research question aims to identify the common software architecture design challenges while developing machine learning systems. A systematic literature review is undertaken to understand better the issues most companies and literature identified during the implementation of ML systems. SLR data are essential in identifying the problems of the overall

analysis of the issue stated in papers. Interviews helped to get a deeper grasp of common challenges from industry practitioners. In addition, Tables 2 and 6 included descriptions of the common challenges posed by both SLR and interviews. I performed a comparison of the SLR results with the interview practitioners. All common challenges discovered by SLR and interviews are mapped into table 9.

The detailed results analysis explanation of SLR of RQ1 is Chapter 6 under "SLR RQ1 Common Challenges Codes, Theme and Description". The interview data analysis of SLR is explained in "5.2.2.1-Interview Analysis of the Software Architecture Design Common Challenges in Machine Learning".

Table 9 mapping represents the common problems discussed in academic literature and by practitioners in the interview. For example, Id number SC2 matches with interview common challenges Id p1-11, which indicates that all practitioners discuss SC2 challenge except one. SC2 common challenge is related to Data management design issues, including the data pre-processing and data preparation in the machine learning system. The data management issue even remained after the ML system was fully developed and deployed in the working environment.

There are other common design challenges discussed in both (SLR, Interview). For example, the usage of microservice architecture common challenge is that it's difficult to find the failure nodes in the ML system. The identification failure in the microservice architecture may add impact, resulting in the failure of the entire system. The evaluation of the ML software architecture design is not reliable because uncertainty can still exist. Another significant common challenge is to gather the experts to identify the appropriate designs for ML systems. The change in input data is a common challenge that can cause the change in the code, model retraining, testing the whole system again, and might impact the final system's outcomes. The wrong selection of the model and not being aware of appropriate design decisions can lead to poor performance of ML systems. The last mapped challenge is related to not having enough knowledge of the development of the ML system. An architect can apply the same design practice of a non-ML system to the ML development workflow, but ML and Non-ML have different processes like requirement, design, testing/quality, process, and management.

The mapped and unmapped can provide guidelines to read all common challenges of SA design in ML systems in a single paper.

6.1.2 SLR-Interview-RQ2 Best Practices Mapping

Id	Best Practices SLR Id	Interview Number Id	Source
SIP1	SP2	P7	SLR, Interview
SIP2	SP3	P3, P5, P11, P12	SLR, Interview
SIP3	SP12	P8	SLR, Interview

SIP4	SP13	P10	SLR (SP17), Interview
SIP5	SP14	P12	SLR, Interview
SIP6	SP15	P8	SLR, Interview
SIP7	SP19	P8	SLR, Interview
SIP8	SP21	P11	SLR, Interview

Table 10 SLR plus Interview Results of RQ2

6.1.2.1 Answer of RQ2

What are the best practices in architecting and designing machine learning systems?
This research question aims to find best practices in architecting and designing machine learning systems. The best design practices can guide researchers and industrial development teams to gain more knowledge based on the needs of their machine learning systems. In an attempt to obtain best practices, the SLR and interviews were performed. The SLR of RQ2 is all results added in table 3, and Interview RQ2 results are mapped in table 7. The detailed results analysis explanation of SLR of RQ2 is in Chapter 6 under "SLR RQ2 Best Practices Codes, Theme and Description". The interview data analysis of SLR is explained in "5.2.3.1- Interview Analysis of the Software Architecture Design Best Practices in Machine Learning". Further, the comparison of SLR results is conducted with the interview outcomes and is mapped in Table 10.

Table 10 mapping represents the matching of best practices in architecting and designing machine learning systems discussed in academic literature and by practitioners in the interview. For example, the best design practice to remove the distortion in scientific image databases is to use the in-memory distributed learning architecture. Another best practice is to use the robust principal component analysis (RPCA) method, which can help to enhance the security of that ML system that is developed using microservice architecture. The RPCA is discussed above in SP3 Id. The next mapped practice is the usage of TensorFlow extended, whose primary strategy is to give users a unified configuration of recurrence components (data analysis and transformation, data validation, model training, model assessment, validation, and infrastructure service). Furthermore, the best design practice that can assist in designing ML systems is the usage of documentation that improves the ML System's reusability. Due to time constraints, a pipeline is often quickly constructed to meet a specific requirement, which can be caused to make a Pipeline jungle. The best practice is to avoid Pipeline jungles by taking a step back, looking at systems as a whole, and taking data collection and feature extraction seriously. The best practice is to use TensorFlow because of its graphical approach; TensorFlow provides a better way of visualizing data. The last mapped best practice is to consider ML and Non-ML system best practices differently

as both follow the different design, implementation, deployment, and maintenance techniques. The best practice while designs an ML system is to distinguished ML and Non/ML systems and take both designing and development techniques separately.

Both mapped and unmapped results of best design practices result can be beneficial in architecting and designing machine learning systems.

6.1.3 SLR-Interview-RQ3 Mian Software Architecture Design Decisions Mapping

Id	Main Decisions Guidelines SLR Id	Interview Number Id	Source
SIM1	SM2	P7	SLR, Interview
SIM2	SM7	P9	SLR, Interview
SIM3	SM13	P8	SLR, Interview
SIM4	SM16	P7	SLR, Interview
SIM5	SM18	P1, P4, P6, P7	SLR, Interview

Table 11 SLR plus Interview Results of RQ3

6.2.3.1 Answer of RQ3

What are the main architectural decisions on Software architecture design for machine learning systems?

The study goal is to find the main software architecture design decisions for machine learning systems. The SLR and interview aim to determine the major decisions in the machine learning system being used. Through SLR total of 18 main architecture decisions have been identified, and interview practitioners also shared some main architecture decisions according to their knowledge. Table 4 shows the SLR of RQ3 outcomes, and table 8 represents the interview RQ3 results. The detailed results analysis explanation of SLR of RQ2 is in Chapter 6 under “SLR RQ3 Mian Software Architecture Design Decisions Codes, Theme, and Description”. The interview data analysis of SLR is explained in “5.2.4.1-Interview Analysis of the Main Software Architecture Design Decision in Machine Learning”. In the end, I mapped both SLR and Interview results in table 11 to discover the common perspective.

Table 11 mapping represents the matching of main software architecture design decisions in machine learning systems discussed in academic literature and by practitioners in the interview. For example, one of the main design decisions is to consider model validation and check the validation results before pushing into the production environment because it is difficult to predict whether a learning algorithm will behave reasonably on new data.

Another design decision is to create a new pattern when available patterns are not feasible for the ML systems because the new pattern can be developed according to system requirements. Furthermore, the main design decision in ML-based safety-critical components is that all safety-critical software components must be written according to strict coding standards and certified by appropriate certification organizations because safety-critical systems failure can cause significant damage to property, environment, or loss of life.

Both mapped results and unmapped can help in making major design decisions in ML systems.

Chapter 7

7
Conclusion and Future Work

7.1 Conclusion

Architectural design decisions have a long-term influence on software system architecture. The development team must carefully consider design decisions from available studies or consult with experienced architects or developers before implementing design selections. This article suggested common design challenges, best design practices, and significant software architecture design decisions for machine learning systems. Data are collected by performing a systematic literature review on four databases and 12 conducted interviews. I identified 34 primary studies for an SLR and the other 7 papers by snowballing. A total of twelve interviews were conducted with experienced machine learning experts from industry and academia.

Further, the SLR outcomes were compared with interviews and mapped in the table. The mapped data represents the standard design decisions recommendation from both sources (SLR and Interviews), indicating a rich SLR results analysis. Both mapped and unmapped data can be beneficial while taking designing for ML systems. I compiled 17 common challenges, 25 best design practices, and 18 major design decisions from SLR for software architecture design decisions in machine learning systems. The 12 interviewee opinions related to research questions are added in tables. As a result, some widespread concerns are discussed in literature and interviews, such as data management, data labeling, and software architecture design models selection while updating the design model with more data. Similarly, frequently occurring challenges, best practices, and some major software architecture design decisions can provide guidelines to overcome issues and help in the decision-making process for ML systems.

This study is useful for machine learning experts, especially those who participated in software architecture design decisions, and it provides the best architecture for ML systems. The study suggests design decisions for different types of ML systems to achieve their business goals. For example, the machine learning applied in image recognition encounters various design difficulties, such as performance, and I developed a list of relevant design decisions to overcome such concerns. Two interviewees highlighted similar common issues related to image processing in ML, and their comments are included in this paper to assist in the design of ML systems.

Five interviewees suggested finding the domain of the ML system before starting the design process; for example, it’s essential to know whether the business requires the system to be deployed as static or dynamic. Furthermore, data management techniques, such as data cleaning, design-related data collection, data quality measurement, and data labeling decisions, help select the best model for its training. Similar problems are discussed in the paper, and further interviewee expert’s opinions are added in tables.

Identifying common, occurring challenges in ML systems and overcoming such issues is not a straightforward process. At the start of the ML system development, the researcher and architects need to find the ML system type which will develop and find all related common challenges, best practices, and main SA design decisions to avoid obstacles that simplify the process. The most prominent issues are data fluctuation over time. While creating a machine learning system is the most critical factor in predicting outcomes according to the input training data. It is complicated when data changes over time and new models must be created regularly. In the following situation, model training and production via component, which is automation, are more design challenging.

Similarly, other most common challenges like a service failure in an MSA (microservice architecture) might have an avalanche effect, resulting in the entire system failing. The MSA application is separated into numerous services, locating the failure nodes is challenging. Another critical challenge in the design of ML occurs when the software with its dynamic behavior, which must be allocated to the non-critical domain according to the applicable design paradigm, must connect with critical applications in the static portion.

The study also found the ML systems best design practices described in the literature and machine learning engineers research and industrial professionals in interviews. For example, the best practices available to design ML systems that handle big data, autonomous systems, digital forensics systems, and safety-critical systems developed using ML. However, I also found some best practices while selecting software architecture models like microservices, TensorFlow, and distributed learning architecture to use in designing machine learning systems. Utilizing this study in the practical field is very beneficial. Some practice focuses on documentation, improving the ML system reusability, and the best way to avoid design smells and debt. The third research question focus on the main software architecture design decision, and these guidelines are coded in tables. I identified many major design decisions that can be used in the design decision process for both static and dynamic ML systems and the issue occurring stage of data corruption. Model incorrectly predicted results could also be avoided by using above mentioned major design decisions.

7.2 Future Work

I intend to expand this study in future work, perform surveys with questionnaires, and arrange a validation workshop with the practitioners for more analysis. I will look at common challenges, best design practices, and main decisions while ML systems are in production and ML components in the integration stage, which will help machine learning engineers gain related knowledge and contribute to the literature.

As a part of future work, I intend to develop a tool for ML teams. Using search string, the development team can find all related best design practices, common challenges, and significant software architecture design decisions available in the literature and collected opinions from ML experts. System users can download excel type documents of extracted results and mainly can filter on which type of ML system they want to develop. While this research is limited to ML, I may look at how well study findings apply to other areas within the more significant subject of AI.

References

[1] S. Amershi, A. Begel, C. Bird, R. DeLine, H. C. Gall, E. Kamar, N. Nagappan, B. Nushi and T. Zimmermann, "Software Engineering for Machine Learning: A Case Study," *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE (SEIP) 2019, Montreal, QC, Canada,,* May 25-31, 2019.

[2] H. Washizaki, H. Uchida, F. Khomh and Y. Gueheneuc, "Studying Software Engineering Patterns for Designing Machine Learning Systems," *2019 10th International Workshop on Empirical Software Engineering in Practice (IWESEP),* no. 30 December 2019, 13-14 Dec. 2019.

[3] A. Apostolos, C. Sofia and S. Loannis, "Research state of the art on GoF design patterns: A mapping study," *Journal of Systems and Software,* p. 1945–1964, July, 2013.

[4] J. Growin, D. Weynys and T. Holvoet, "Design Patterns for Multi-agent Systems: A Systematic Literature Review," *Agent-Oriented Software Engineering: Reflections on Architecture's, Methodologie's, Language's, and Framework's,* pp. 79-99, 02 2014.

[5] A. Hany, W. Abedelmoez and S. Hamdi, "Software Engineering Using Artificial Intelligence Techniques: Current State and Open Problems," March, 2012.

[6] S. Shirwaikar, "An Exploratory Study of the Security Design Pattern Landscape and their Classification," *International Journal of Secure Software Engineering,* vol. 7, pp. 26-43, 07 2016.

[7] Y. Watanabe, H. Washizaki, K. Sakamoto, D. Saito, K. Honda, N. Tsuda, Y. Fukazawa and N. Yoshioka, "Preliminary Systematic Literature Review of Machine Learning System Development Process," 10 2019.

[8] B. Yazdi, M. Bafandeh, A. Rasoolzadegan and A. Ghavidel, "The state of the art on design patterns: A systematic mapping of the literature," *Journal of Systems and Software,* vol. 125, pp. 93-118, 2017.

[9] H. Liu, S. Eksmo, J. Risberg and R. Hebig, "Emerging and Changing Tasks in the Development Process for Machine Learning Systems," vol. 125–134, 2020.

[10] T. Al-Naeem, F. T. Dabous, F. A. Rabhi and B. Benatallah, "Formulating the Architectural Design of Enterprise Applications as a Search Problem," *Proceedings of the 2005 Australian Conference on Software Engineering,* p. 282–291, 2005.

[11] E. M. Saleh, O. Sallabi and H. A. Darbi, "A Multi-Agent System to Support Design Pattern Recommendation," 2020.

[12] S. Schelter, B. F, T. Januschowski, D. Salinas and S. S. G. Seufert, "On Challenges in Machine Learning Model Management," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering,* pp. 1-11, 2018.

[13] P. Zdun and U. Avgeriou, “Architectural patterns revisited - A pattern Language,” *EuroPLoP’ 2005, Tenth European Conference on Pattern Languages of Programs, Irsee, Germany,* pp. 431-470, July 6-10, 2005,.

[14] H. Washizaki, H. Takeuchi, F. Khomh, N. Natori, T. Doi and S. Okuda, “Practitioners’ insights on machine-learning software engineering design patterns: a preliminary study},” *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME),* pp. 797-799, 2020.

[15] H. Washizaki, N. Yoshioka, A. Hazeyama, T. Kato, H. Kaiya, S. Ogata, T. Okubo and E. Fernandez, “Landscape of IoT Patterns, Proceedings of the 1st International Workshop on Software Engineering Research & Practices for the Internet of Things,” *IEEE Press,* 2019.

[16] J. Wang, G. Li and Y. Pu, “A Scenario-Based Architecture for Reliability Design of Artificial Intelligent Software,” pp. 6-9, 2010.

[17] R. Mayer and H.-A. Jacobsen, “Scalable Deep Learning on Distributed Infrastructures: Challenges, Techniques, and Tools,” *ACM Computing Surveys (CSUR),* vol. 53, pp. 1-37, 02 2020.

[18] Z. a. X. X. a. L. D. a. M. G. C. Wan, “How does Machine Learning Change Software Development Practices?,” *IEEE Transactions on Software Engineering,* pp. 1-1, 2019.

[19] H. Muccini and K. Vaidhyanathan, “Software Architecture for ML-based Systems: What Exists and What Lies Ahead,” 03 2021.

[20] H. Yokoyama, “Machine Learning System Architectural Pattern for Improving Operational Stability,” *2019 IEEE International Conference on Software Architecture Companion (ICSA-C),* pp. 267-274, 2019.

[21] B. Kitchenham and S. Charters, “Guidelines for performing Systematic Literature Reviews in Software Engineering,” vol. 2, 01 2007.

[22] P. Regnell, H. Martin and A. Rainer, “Case Study Research in Software Engineering – Guidelines and Examples,” 2012.

[23] D. Amaratunga, D. Baldry, M. Sarshar and R. Newton, “Quantitative and qualitative research in the built environment: Application of 'mixed' research approach,” vol. 51, pp. 17-31, 02 2002.

[24] C. Okoli, “A Guide to Conducting a Standalones Systematics Literatures Reviews,” *Communications of the Association for Information Systems,* vol. 37, 2015.

[25] O. Ulrika, L. Kidd, Y. Wengström and N. Rowa-Dewar, “Combining qualitative and quantitative research within mixed method research designs: A methodological review,” *International journal of nursing studies,* 11 2010.

[26] T. Meline, “Selecting Studies for Systemic Review: Inclusion and Exclusion Criteria,” *Contemporary Issues in Communication Science and Disorders,* vol. 33, pp. 21-27, 03 2006.

[27] C. Wohlin, “Guidelines for snowballing in systematic literature studies and a replication in software engineering,” *in Proceedings of the 18th international conference on evaluation and assessment in software engineering,* pp. 1-10, 2014.

[28] J. Webster and R. Walton T., “Analyzing the past to prepare for the future: Writing a literature review.,” *MIS quarterly,* pp. xiii-xxiii, 2002.

[29] T. a. C. M. a. F. B. a. J. I. Punter, “Conducting on-line surveys in software engineering,” *2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings.,* pp. 80-88, 2003.

[30] K. Dillenburger, “Survey Research Methods, Floyd J. Fowler Jnr, London, Thousand Oaks, CA, Sage Publications, 3rd edn, 2002, pp. ix + 179, Cloth ISBN 0 7619 2190 7, pound38.00, Paper ISBN 0 7619 2191 5, pound14.99,” *British Journal of Social Work - BRIT J SOC WORK,* vol. 32, pp. 390-391, 2002.

[31] F. Shull, J. Singer and D. I. Sjøberg, “Guide to Advanced Empirical Software,” 2007.

[32] G. Hackett, “Survey research methods,” *Personnel Guidance Journal,* vol. 59, p. 9, 1981.

[33] C. Wohlin, R. Per, H. Matin, M. C. O, R. Björn and W. Anders, “Empirical Strategies." In Experimentation in Software Engineering,” no. Springer, Berlin, Heidelberg, pp. 9-36, 2012.

[34] X. Zhou, Y. Jin, H. Zhang, S. Li and X. Huang, “A map of threats to validity of systematic literature reviews in software engineering,” *Proceedings of the 23rd Asia-Pacific Software Engineering Conference,* pp. 153-160, Dec, 2016.

[35] M. Kassab, J. DeFranco and P. Laplante, “Software Testing: The State of the Practice,” *IEEE Software,* pp. 46-52, 2017.

[36] A. Nguyen-Duc, I. Sundb, N. Elizamary, C. Tayana, A. Iftekhar and A. Pekka, “A Multiple Case Study of Artificial Intelligent System Development in Industry,” *EASE '20: Evaluation and Assessment in Software Engineering Trondheim Norway,* p. 1–10, 04 2020.

[37] J. McGovern, S. W. Ambler, M. Stevens, J. Linn, E. K. Jo and V. Sharan, “The Practical Guide to Enterprise Architecture,” 2003.

Appendix A

Primary Studies for SLR:

S1) J. Wang, G. Li and Y. Pu, "A Scenario-Based Architecture for Reliability Design of Artificial Intelligent Software," International Conference on Computational Intelligence and Security, pp. 6-9, 2010.

S2) A. Serban, K. van der Blom, H. Hoos and J. Visser, "Adoption and Effects of Software Engineering Best Practices in Machine Learning," In: Proceedings of the 14th International Symposium on Empirical Software Engineering and Measurement, pp. 1-12, 2020.

S3) J. Schleier-Smith, "An Architecture for Agile Machine Learning in Real-Time Applications," 15: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p. 2059–2068, 2015.

S4) Z. Wan, X. Xia, D. Lo and G. C. Murphy, "How does Machine Learning Change Software Development Practices?," IEEE Transactions on Software Engineering, pp. 1-1, 2019.

S5) J. Musil, A. Musil and S. Biffl, "Introduction and Challenges of Environment Architectures for Collective Intelligence Systems," Agent Environments for Multi-Agent Systems IV, vol. 9068, pp. 76-94, 2015.

S6) H. Yokoyama, "Machine Learning System Architectural Pattern for Improving Operational Stability," IEEE International Conference on Software Architecture Workshops (ICSAW), pp. 267-274, 2019.

S7) A. Musil, J. Musil and S. Biffl, "Major Variants of the SIS Architecture Pattern for Collective Intelligence Systems," Proceedings of the 21st European Conference on Pattern Languages of Programs, pp. 1-11, 2016.

S8) S. e. a. Amershi, "Software Engineering for Machine Learning: A Case Study," 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), pp. 291-300, 2019.

S9) H. Washizaki, H. Takeuchi, F. Khomh, N. D. T. Natori and S. Okuda, "Practitioners' insights on machine-learning software engineering design patterns: a preliminary study," 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 797-799, 2020.

S10) R. Mayer and H. Jacobsen, "Scalable Deep Learning on Distributed Infrastructures: Challenges, Techniques, and Tools," ACM Computing Surveys (CSUR), vol. 53, no. 1, pp. 1-37, 2020.

S11) H. Muccini and K. Vaidhyanathan, "Software Architecture for ML-based Systems: What Exists and What Lies Ahead," arXiv:2103.07950, pp. 1-8, 2021.

S12) A. Ahmad and M. Babar, "Software architectures for robotic systems: A systematic mapping study," Journal of Systems and Software, vol. 122, pp. 16-39, 2016.

S13) H. Washizaki, H. Uchida, F. Khomh and Y. Guéhéneuc, "Studying Software Engineering Patterns for Designing Machine Learning Systems," 2019 10th International Workshop on Empirical Software Engineering in Practice (IWESEP), pp. 49-495, 2019.

S14) M. Scheerer, J. Klamroth, R. Reussner and B. Beckert, "Towards Classes of Architectural Dependability Assurance for machine-learning-based systems," SEAMS '20: Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pp. 31-37, 2020.

S15) C. Castellanos, B. Pérez, D. Correal and C. Varela, "A Model-Driven Architectural Design Method for Big Data Analytics Applications," 2020 IEEE International Conference on Software Architecture Companion (ICSA-C), pp. 89-94, 2020.

S16) A. Biondi, F. Nesti, G. Cicero, D. Casini and G. Buttazzo, "A Safe, Secure, and Predictable Software Architecture for Deep Learning in Safety-Critical Systems," IEEE Embedded Systems Letters, vol. 12, no. 3, pp. 78-82, 2020.

S17) A. Panousopoulou, S. Farrens, K. Fotiadou, A. Woiselle, G. Tsagkatakis, J. Starck and P. Tsakalide, "A Distributed Learning Architecture for Scientific Imaging Problems," arXiv:1809.05956 , pp. 1-40, 2018.

S18) M. e. a. Jin, "An Anomaly Detection Algorithm for Microservice Architecture Based on Robust Principal Component Analysis," IEEE Access, vol. 8, pp. 226397-226408, 2020.

S19) P. Alarcon, M. Gomez, J. Campos, S. Aguilar, S. Romero and P. Serrahima, "A Holistic Approach for Intelligent Automated Control Technology for Balanced Automation Systems," The International Federation for Information , pp. 301-308, 1995.

S20) L. Spalazzi, M. Paolanti and E. Frontoni, "An offline parallel architecture for forensic multimedia classification," Multimedia Tools and Applications, 2021.

S21) A. e. a. Berquand, "Artificial Intelligence for the Early Design Phases of Space Missions," 2019 IEEE Aerospace Conference, pp. 1-20, 2019.

S22) A. Serban, "Designing Safety Critical Software Systems to Manage Inherent Uncertainty," 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), pp. 246-249, 2019.

S23) V. Indumathi and G. Nasira, "FAULT TOLERANCE IN JOB SCHEDULING THROUGH FAULT MANAGEMENT FRAMEWORK USING SOA IN GRID," ICTACT JOURNAL ON SOFT COMPUTING, vol. 07, no. 02, pp. 1-5, 2017.

S24) L. Li, J. Wang and C. Xu, "FLSim: An Extensible and Reusable Simulation Framework for Federated Learning," ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 350-369, 2021.

S25) O. Fomin and O. Derevianchenko, "Improvement of the Quality of Cutting Tools States Recognition Using Cloud Technologies," Advances in Design, Simulation and Manufacturing III. DSMIE 2020, pp. 243-252, 2020.

S26) E. Kusmenko, S. Nickels, S. Pavlitskaya, B. Rumpe and T. T, "Modeling and Training of Neural Processing Systems," 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS), pp. 283-293, 2019.

S27) M. e. a. Möstl, "Platform-Centric Self-Awareness as a Key Enabler for Controlling Changes in CPS," Proceedings of the IEEE, vol. 106, no. 9, pp. 1543-1567, 2018.

S28) A. Muzaffar, S. Mir, M. Latif, W. Butt and A. F, "Software Architecture of a Mobile Robot," International Conference on Computational Science and Computational Intelligence, pp. 1-6, 2015.

S29) B. Vinayagasundaram and S. Srivatsa, "Software quality in artificial intelligence system," Information Technology Journal, vol. 6, pp. 835-842, 2007.

S30) A. Serban, E. Poll and J. Visser, "Towards Using Probabilistic Models to Design Software Systems with Inherent Uncertainty," European Conference on Software Architecture ECSA 2020 Software Architecture , pp. 89-97, 2020.

S31) E. Buccio, A. Lorenzet, M. Melucci and F. Neresini, "Unveiling Latent States Behind Social Indicators".

S32) M. Bhat, K. Shumaiev, K. Koch, U. Hohenstein, A. Biesdorf and F. Matthes, "An expert recommendation system for design decision making: Who Should be Involved in Making a Design Decision?," 2018 IEEE International Conference on Software Architecture (ICSA), pp. 85-8509, 2018.

S33) H. Venthur, S. Dähne, J. Hönhe, H. Heller and B. Blankertz, "Wyrm: A Brain-Computer Interface Toolbox in Python," Neuroinformatics, vol. 13, p. 471–486, 2015.

S34) D. Baylor and et al., "TFX: A TensorFlow-Based Production-Scale Machine Learning Platform," Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p. 1387–1395, 2017.

S35) A. Anjos, M. Günther, T. de Freitas Pereira, P. M. A. Korshunov and S. Marcel, "Continuously Reproducing Toolchains in Pattern Recognition and Machine Learning Experiments," ICML 2017 RML Program Chairs, 2017.

S36) B. Burns and D. Oppenheimer, "Design patterns for container-based distributed systems," USENIX} Association, Denver, 2016.

S37) P. Raut and N. Borkar, "Machine Learning Algorithms:Trends, Perspectives and Prospects," International Journal of Engineering Science and Computing, vol. 7, no. 3, pp. 4884-4891, 2017.

S38) D. e. a. Sculley, "Hidden Technical Debt in Machine Learning Systems," Advances in Neural Information Processing Systems, Curran Associatesd, Inc.,, pp. 1-9, 2015.

S39) S. Schelter, B. F, T. Januschowski, D. Salinas and S. S. G. Seufert, "On Challenges in Machine Learning Model Management," Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, pp. 1-11, 2018.

S40) M. e. a. Abadi, "TensorFlow: A System for Large-Scale Machine Learning," THE ADVANCE COMPUTING SYSTEMS ASSOCIATION, pp. 1-21, 2016.

S41) N. Chervyakov, P. Lyakhov, M. Deryabin, N. Nagornov, M. Valueva and G. Valuev, "Residue Number System-Based Solution for Reducing the Hardware Cost of a Convolutional Neural Network," Neurocomputing, vol. 407, pp. 439-453, 2020.

Appendix B

Interview Invitation:

INTERVIEW INVITATION

INVITATION TO PARTICIPATE IN OUR RESEARCH PROJECT TITLED "SOFTWARE ARCHITECTURE DESIGN CHALLENGES FOR MACHINE LEARNING SYSTEMS"

Hello

My name is Roger Nazir. I am doing my Master Thesis in Software Engineering under the guidance of Associate Professor Patrizio Pelliccione from Chalmers University of Technology, Sweden. This research aims to investigate architectural design decisions for Machine Learning (ML) systems. This study aims to help developers to have a comprehensive and order classification of common challenges, best practices, and main software architecture (SA) design decisions of ML systems from the available studies. It will also highlight the ML software design complexity and common ML design techniques. The study will help developers/designers to learn the best ML design practice to minimize the challenges in creating large-scale ML solutions. I hope this study will offer some essential contributions to future research work and present a platform to assist young architects by suggesting appropriate architecture designs. Initially, I am conducting a systemic literature review with snowballing. Then through an In-depth interview, I will collect the practitioner's opinions, which will be compared to the results of the systematic literature review.

I need your time for interviewing a video call, which takes 30-40 minutes. Your participation in this study based on your knowledge and experience will be valuable to our research. The interview session will only be recorded after getting permission from the interviewee. No such details will be added in the study that will point to any personal information of the interviewee. We kindly invite you to give your opinions if you are willing to participate. Please suggest a day and time that suits you, and I will do my best to be available. If you have any questions, please do not hesitate to ask. I appreciate any help you can provide.

Regards,
Roger Nazir

SUPERVISOR:
Associate Professor Patrizio Pelliccione
Department of Computer Science and Engineering Software Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
patrizio.pelliccione@cse.gu.se

INTERVIEW QUESTIONNAIRE

STUDYING SOFTWARE ARCHITECTURE DESIGN CHALLENGES FOR MACHINE LEARNING SYSTEMS

Roger Nazir
SUPERVISOR:
ASSOCIATE PROFESSOR PATRIZIO PELLICCIONE
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SOFTWARE ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY

Questions:

1. Can you please introduce yourself and describe your job role in this company?

2. Since how many years you are working in this company?

3. Have you published any thesis in the machine learning domain?

4. Can you please share your experience in your current position?

5. Do you have any experience in the previous company which is developing Machine Learning system? If so, then what was your old experience?

6. Is your company is service-based or product-based?

7. What software development model do you practice in your company in general, like an agile, waterfall, etc.?

8. Could you please share your experience with the interesting machine learning project you have recently worked on?

9. In your working experience, how many software architecture design techniques of machine learning you worked with?

10. Which common software architecture design technique of machine learning you found being used in most companies through your experience?

11. According to your experience, which are your best software architecture design technique for machine learning, and what are the benefits of using them?

12. Do you have any recommendations for software architecture design techniques of machine learning systems?

13. Which would be the best practice that could be useful/helpful in applying software architecture designing of machine learning systems?

14. What are the most common software architecture design challenges in machine learning systems?

15. What are the main architectural decisions on software architecture design of different machine learning systems?