What Do We Know About Operationalizing Machine Learning Models? A Systematic Literature Review.

Ask Berstad Kolltveit

December 14, 2021

Abstract

Deploying machine learning (ML) models to production with the same level of rigor and automation as traditional software systems has shown itself to be a non-trivial task, requiring extra care and infrastructure to deal with the additional challenges. This study is a systematic literature review investigating the state of the art in operationalizing ML models using the following research questions.

- **RQ1: How are ML models operationalized in the state of the art?**
- **RQ2: What are the main challenges in operationalizing ML models?**
- **RQ3: What tools and software infrastructure are used to operationalize ML models?**
- **RQ4: What are the feature gaps in the tooling and infrastructure used to operationalize ML models?**

The results of the review show that there are many different techniques for operationalizing ML models, and there exists a number of tools for the most common use cases, cloud deployment in particular. The review also revealed several opportunities for research into better tooling and infrastructure, particularly for serving and edge ML deployments. The results of the study should provide a better overview of different deployment approaches and architectures than previous reviews on the topic, as well as giving an overview of some of the tooling used and opportunities for future research.

Contents

Figures

Tables

Acronyms

AI artificial intelligence. 3, 6, 29

API application programming interface. 17, 18, 21

CD continuous deployment. 5, 6

CPU central processing unit. 23

GL grey literature. 6, 28

GLR grey literature review. 6, 28, 29

GPU graphics processing unit. 24

IO input/output. 25

IoT Internet of Things. 20, 21, 24

KA knowledge area. 1, 5

KPI key performance indicator. 19

ML machine learning. iii, 1–7, 9–11, 17–24, 27, 29

MLaaS machine learning-as-a-service. 23

MLR multivocal literature review. 6, 29

REST representational state transfer. 17, 18, 21

RPC remote procedure call. 17, 18, 21

RQ research question. 1

SE software engineering. 1–6, 9, 10, 28

SLO service level objective. 18, 20, 22, 23

SLR systematic literature review. ix, 1, 5, 6, 9–11, 27–29

SMS systematic mapping study. 5, 6

TPU tensor processing unit. 24

Chapter 1

Introduction

Machine learning models have become increasingly prevalent in virtually all fields of business and research in the past decade, with a steady stream of new algorithms and techniques being published. However, deploying models to production with the same level of rigor and automation as traditional software systems has shown itself to be a non-trivial task. As a result, MLOps — an extension of DevOps to include machine learning (ML) systems — and software engineering (SE) for ML in general, is experiencing increased attention from researchers and practitioners.

In recent years, several literature reviews have been conducted to investigate the intersection of SE and ML, mainly mapping practices and challenges to knowledge areas (KAs) of the SWEBOK [1] such as in [2] and [3]. In this review, the operationalization of ML models and the surrounding infrastructure is the subject of investigation. This entails everything that happens after a model has been trained and evaluated, from packaging and integration, to serving and monitoring. The different approaches reported in literature will be described, together with tools used and challenges often encountered. In addition, reported gaps in current tooling will also be investigated in order to provide opportunities for future contributions.

This study will attempt to answer the following research questions (RQs).

- **RQ1: How are ML models operationalized in the state of the art?**
- **RQ2: What are the main challenges in operationalizing ML models?**
- **RQ3: What tools and software infrastructure are used to operationalize ML models?**
- **RQ4: What are the feature gaps in the tooling and infrastructure used to operationalize ML models?**

In order to answer the RQs, a systematic literature review (SLR) was conducted based on the guidelines of [4] and [5]. One iteration of forward and backward snowballing was performed on a start set of 19 studies, resulting in 24 papers to review after performing study selection and quality assessment. The results of the review show that there are many different techniques for operationalizing ML

models, and there is an abundance of tooling available for the most common use cases, cloud deployment in particular. The review also revealed several opportunities for research into better tooling and infrastructure, particularly for serving and edge/distributed ML deployments. The results of the study should provide a better overview of different deployment approaches and architectures than previous reviews on the topic. Furthermore, the study gives an overview of some of the tooling used, as well as opportunities for future research.

The study is organized as follows. A brief background on ML and DevOps is given in Chapter 2. A summary of and comparison with earlier literature reviews in the field of SE for ML is found in Chapter 3. The research methodology and corresponding implementation are described in Chapter 4. The results of the study are presented in Chapter 5. Chapter 6 compares the results of this study with previous work, as well as discussing some limitations. Finally, some directions for future work and research are proposed in Chapter 7

Chapter 2

Background

This chapter will briefly explain the state of machine learning today, operationalization of ML systems and motivate the need for MLOps.

2.1 Machine Learning

ML models have become increasingly prevalent in virtually all fields of business and research in the past decade. *The state of AI in 2020* [6] reports that 50% of businesses surveyed have adopted artificial intelligence (AI) in some business function. New algorithms and techniques for learning, optimizing and improving models are being intensely researched, with new milestones achieved year after year. With all the resarch that has been done on the training and evaluation of machine learning models, the difficulty for most companies and practitioners now is not to find new algorithms and optimizations in training, but rather how to actually deploy models to production in order to delived tangible business value. Most companies are still in the very early stages of incorporating ML in their business processes [7].

2.2 Software Engineering for ML Systems

While traditional SE for non-ML systems is a mature and well-understood field, software engineering for ML systems is still a young and immature knowledge area. Traditional software systems are largely deterministic, computing-driven systems whose behavior is purely code-dependent. On the other hand, ML models have an additional data dependency, in the sense that their behavior is learned from data, and they have even been characterized as non-deterministic [8, 9]. The additional data dependency is one of the factors contributing to the fact that ML systems require a great amount of supporting infrastructure, leading to an accretion of significantly more technical debt than traditional software systems, for example in the form of entanglement, unstable data dependencies or glue code [10]. As a result of the extra technical debt, ML systems are more challenging to

deploy, something that the research community has recently turned its attention towards.

2.3 DevOps for ML Systems – *MLOps*

DevOps is a subset of software engineering focused on tightening the coupling between development and operation of software systems. DevOps principles advocate for end-to-end automation [11], which is expressed through the use of version control systems, automated build and deploy pipelines, etc. Some motivating factors for automation are shortening the time to delivery, increasing reproducibility and reducing time spent on automatable processes [12]. DevOps for Machine Learning, named MLOps, is a subset of SE for ML and a superset/extension of DevOps, focused on adopting DevOps practices when developing and operating ML systems [13]. This is because existing DevOps practices are not sufficient for ML systems, which pose additional requirements. ML systems not only have code dependencies, but have data dependencies in addition, which may impose the requirement of data monitoring (for data distributions shift), continuous training, automatic retraining, etc.

2.4 Working Definition of Operationalizing ML Systems

Operationalization consists of taking the system from a development environment (e.g. local development machine) to a production environment (e.g. a server), and in the context of this paper will encompass all steps that come after model training and evaluation, which typically includes (but is not restricted to): packaging model in a format appropriate for deployment, publishing to a model registry or model storage, integration into a broader software system, serving, and monitoring.

Chapter 3

Related Work

Shahin *et al.* [14] performed a comprehensive systematic literature review (SLR) of tools, practices, approaches and challenges for continuous SE. The authors identified thirty approaches and associated tools for facilitating continuous SE practices, and found they were being applied in a wide variety of application domains. The application of the identified approaches to ML systems is not discussed or mentioned.

Rodríguez *et al.* [15] performed a systematic mapping study (SMS) of continuous deployment (CD), focusing on the state of research, characteristics of CD and benefits, challenges and gaps in the research. The authors identify nine concrete areas for future research opportunities, but the topic of applying CD to ML is not discussed.

Baier *et al.* [16] surveyed the academic literature for challenges in deploying ML models. The authors also conducted eleven semi-structured expert interview with ML practitioners from a wide variety of industries. The data from the literature survey and expert interviews were combined and analyzed in order to produce an overview of practical challenges surrounding ML within the categories of pre-deployment, deployment, and non-technical issues. The authors found that the practitioners' experiences generally confirmed challenges reported in the academic literature.

Paleyes *et al.* [17] did a survey of case studies on deploying ML models with the goal of outlining a research agenda for addressing the challenges that practitioners were found to encounter. The authors identified practical challenges in sixteen different steps of ML deployment, and suggest further research to be done in the areas of tooling and services for individual challenges, as well as new holistic approaches for dealing with ML systems engineering.

Kumeno [2] performed an SLR of SE challenges for ML, and mapped them to the twelve knowledge areas (KAs) of the SWEBOK [1]. The author found that challenges in SE for ML are found in all KAs, and that safety/security and non-technical areas in particular have received a lot of attention in the literature.

Nascimento *et al.* [3] performed an SLR of challenges and practices in SE for ML. The challenges and practices are mapped to the KAs of the SWEBOK. The au-

thors found that studies from laboratory environments are mainly concerned with building and testing models, which could indicate that more literature produced by industry practitioners should be studied when looking to review the state of the art and state of practice of deploying models to production. This is further illustrated when the authors highlight that deployment is one of the areas with the fewest suggested practices in published literature.

Lwakatare *et al.* [18] conducted an SLR on SE for ML using a two-dimensional scheme, categorizing challenges according to quality attributes (adaptability, scalability, privacy and safety) and ML development workflow step (data acquisition, training, evaluation, deployment). The authors also identify solutions to some of the challenges, but note that areas such as privacy and safety are missing solutions, as well as areas such as evaluation and deployment. The exclusion of grey literature (GL) is mentioned as a threat to validity of the study.

Lwakatare *et al.* [19] did an exploratory case study on CD for ML. The authors also performed an MLR to create a five-step process improvement conceptual model for the ML deployment process. The results were validated by conducting a focus group with ten ML practitioners at a telecommunications company.

Serban *et al.* [20] identified 29 best practices of SE for ML through conducting an MLR, before conducting a practitioner survey to investigate adoption and impact of the practices. The authors perform extensive quantitative analysis of the survey responses, investigating the relationships between practice and effect, correlation between practices, and the importance of each practice.

John *et al.* [21] performed a multivocal literature review (MLR) of the SE lifecycle of ML models, outlining challenges and best practices in each step of the lifecycle.

Giray [8] conducted an SLR of SE for ML, where challenges and solutions were grouped by areas suggested in SWEBOK, similarly to [2] and [3], while including an even greater number of papers than any previous SLR in the field.

Lorenzoni *et al.* [22] conducted an SLR of ML model development, identifying phases, techniques, gaps and trends in the ML development lifecycle. The deployment aspect is only briefly mentioned.

Martínez-Fernández *et al.* [9] performed an extensive SMS of SE for AI, investigating reported SE approaches and challenges for developing AI systems.

Serban and Visser [23] studied software architecture for machine learning, conducting an SLR and practitioner interviews to find architectural challenges and solutions in ML, while validating the findings with a survey.

John *et al.* [24] performed an SLR and grey literature review (GLR) on the adoption of MLOps practices and how companies evolve through different stages of adoption. A framework is produced, allowing the characterization of a company's MLOps adoption maturity and providing steps on how to improve adoption, similarly to [19].

A common characteristic of previous work is that it tends to be quite broad in scope, covering many areas of SE for ML. This study will make the following novel contributions to the research.

- Focus specifically on how ML models are operationalized, including the aspects of serving and monitoring, which have largely been overlooked in previous work
- Investigate how tooling and infrastructure are used to operationalize models
- Identify reported gaps and challenges related to tools and infrastructure for operationalization

A comparison between this study and the discussed related work with respect to the above listed novel contributions can be found in Table 3.1.

Study	O	TI	R	SE4ML
Shahin *et al.* [14]	X	X		
Rodríguez *et al.* [15]	X			
Baier *et al.* [16]	X			X
Paleyes *et al.* [17]	X			X
Kumeno [2]				X
Nascimento *et al.* [3]				X
Lwakatare *et al.* [18]				X
Lwakatare *et al.* [19]				X
Serban *et al.* [20]				X
John *et al.* [21]	X			X
Giray [8]				X
Lorenzoni *et al.* [22]				X
Martínez-Fernández *et al.* [9]				X
Serban and Visser [23]				X
John *et al.* [24]				X
This study	X	X	X	X

Table 3.1: Comparison of related work with this study. O — in-depth discussion of operationalization; TI — investigates tooling and infrastructure issues; R — identifies reported suggestions for further research in operationalization tools and infrastructure; SE4ML — concerns the application of SE to ML.

Chapter 4

Research Design and Implementation

This chapter will discuss the motivation, design and implementation of the SLR. The chapter is structured as follows. In Section 4.1 the impact and usefulness of the study is argued. Section 4.2 presents the research questions of the study. Section 4.3 outlines the research method and design of the study. Finally, Section 4.4 accounts for how the study was conducted.

4.1 Research Motivation

As outlined in Chapter 2, industry adoption of ML is still in its early stages and rapidly growing. The SE aspect of ML is an active field of research, with the vast majority of work having been done only in the past five years. As reported by earlier literature studies in the field (see Chapter 3), operationalization of ML is an area which presents practitioners with real challenges, such as how to deliver model artifacts to the production environment or how to monitor evolving input data [17]. Tackling operationalization challenges requires adopting good practices as well as utilizing suitable tooling. Earlier work has largely had a broad scope, with goals of mapping out general SE challenges and practices for ML. In short, this study is motivated by two key factors.

- Researching ML operationalization in-depth, and not as part of a broader study of SE for ML.
- Putting more focus on tooling and infrastructure by identifying how current tools are used and what is reported to need further research.

4.2 Research Questions

The overarching research question of this study is "**What is the state of the art in ML deployment?**". The focus will be on tooling and trying to identify feature

gaps reported in studies where model deployment is discussed. To support the main research question, four subquestions have been defined:

- **RQ1: How are ML models operationalized in the state of the art?**
- **RQ2: What are the main challenges in operationalizing ML models?**
- **RQ3: What tools and software infrastructure are used to operationalize ML models?**
- **RQ4: What are the feature gaps in the tooling and infrastructure used to operationalize ML models?**

The working definition of *operationalization* in this paper is outlined in Section 2.4.

4.3 Research Method and Design

This study is a systematic literature review (SLR) based on the guidelines of Kitchenham and Charters [4] and Wohlin [5]. In this section, the research methodology, i.e. review protocol, will be described.

4.3.1 Search Strategy

Typically, candidate papers in an SLR are identified by constructing a search string for querying digital libraries. However, initial string-based searches (using strings such as "machine learning deployment", "MLOps deployment", etc) in Oria in late September 2021 failed to produce adequately relevant articles for the topic at hand. The articles found did not report deployment details related to the operationalization aspect of SE for ML that this study is concerned with. This motivated the use of snowballing as an alternative approach for identifying candidate papers. The snowballing procedure followed the process proposed by Wohlin [5], with the exception that forward and backward snowballing were limited to a single iteration each. Forward snowballing will be conducted using the *Cited by* functionality of Google Scholar[1].

4.3.2 Study Selection

Studies will be selected based on the criteria found in Table 4.1. Inclusion requires that the study satisfies all of the inclusion criteria, while fulfilling only one of the exclusion criteria leads to exclusion. The year 2015 was chosen as earliest year of publication as this was when research on SE for ML began to gain traction [2], possibly in part due to Sculley *et al.* [10].

4.3.3 Study Quality Assessment

The quality of included studies will be assessed using the criteria found in Table 4.2, adapted from criteria used by [8] and suggested by [25]. Studies are

[1] `https://scholar.google.com/`

Inclusion criteria	Written in English Published in a peer-reviewed journal, conference or workshop Published after 2015 Discusses one of the following aspects of ML operationalization: challenges, solutions, tooling, processes, requirements Available online
Exclusion criteria	One of the inclusion criteria is not satisfied

Table 4.1: Study selection criteria used in the SLR.

Criteria	Empirical	Non-empirical
Does the author have authority on the subject?	X	X
Does the source have a clearly stated aim?	X	X
Is the author free of any vested interests?	X	X
Have key related sources been linked to or discussed?	X	X
Is relevance (to industry or academia) discussed?	X	X
Does the source have a stated methodology?	X	
Are any threats to validity clearly stated?	X	

Table 4.2: Study quality assessment criteria for empirical and non-empirical sources, indicated by *X*. Based on criteria from [25] and [8].

subject to assessment criteria marked with *X* depending on whether or not they are empirical. For each criterion, the study is awarded 1 point for *yes*, 0.5 for *partly* and 0 for *no*. Sources with an average of 0.5 points or less are excluded.

4.3.4 Data Extraction

In order to answer the research questions, sufficient data must be gathered from the literature. Thus, the data extraction form found in Table 4.3 was designed. Each field is marked by the research question(s) they are intended to facilitate in answering.

4.3.5 Data Synthesis

Data synthesis will be done through the use of thematic analysis as described in [26]. A simplified version of the procedure will be used based on the relatively small dataset and the researcher's inexperience with qualitative methods. A single cycle of coding will be performed (rather than multiple), followed by reviewing possible themes and finally attempting to articulate the findings.

Field	Explanation
Author	Name of author(s)
Publication year	Year of publication
Context	Description of context where necessary
Achievement	Main achievement of the study
Process	Description of the operationalization process (RQ1)
Tools & infrastructure	Tools and infrastructure used for operationalization (RQ1, RQ3)
Challenges	Operationalization challenges reported (RQ2)
Solutions	Implemented or proposed solutions to reported problems (RQ1, RQ2)
Requirements	Reported requirements for operationalization process, tooling and infrastructure (RQ4)
Gaps	Reported unmet requirements, open problems and proposed research directions in infrastructure or tooling (RQ4)
Other	Any other notes

Table 4.3: Data extraction form with clarification of the content for each field.

4.4 Research Implementation

Selection, quality assessment and data extraction was conducted in an online spreadsheet[2].

4.4.1 Search

In order to start the snowballing, an initial set of studies was assembled. The studies were mainly found through the reference lists of earlier related work, such as [21] and [9]. Some manual search with Google Scholar was also done using search strings such as "Mlops deployment" and "machine learning deployment". The start set can be found in Table A.1.

The study selection criteria found in Table 4.1 were applied to the start set. From the start set of 19 papers, 8 satisfied the selection criteria and minimum study quality score, and were thus selected for review and further snowballing. The papers selected from the start set can be found in Table 4.4.

Backward snowballing from the 8 selected papers provided an additional 21 candidate papers based on skimming title/abstract/content of all referred papers from the start set. Of the 21 candidate papers, 6 papers satisfied the selection criteria and minimum study quality criteria. The papers selected from backward snowballing can be found in Table 4.5.

[2] https://docs.google.com/spreadsheets/d/1VMBLQZBsc5dgldnDfHH8cHAWk9ikjxX0TVq4lfYbMpg/edit?usp=sharing

Study	Title
Hazelwood *et al.* [27]	"Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective"
Hummer *et al.* [28]	"ModelOps: Cloud-Based Lifecycle Management for Reliable and Trusted AI"
Krishnamurthi *et al.* [29]	"Deploying Deep Learning Models via IOT Deployment Tools"
Liu *et al.* [30]	"Building A Platform for Machine Learning Operations from Open Source Frameworks"
Chen *et al.* [31]	"Developments in MLflow"
Bosch *et al.* [32]	"Engineering AI Systems"
Ruf *et al.* [33]	"Demystifying MLOps and Presenting a Recipe for the Selection of Open-Source Tools"
Granlund *et al.* [34]	"Towards Regulatory-Compliant MLOps: Oravizio's Journey from a Machine Learning Experiment to a Deployed Certified Medical Product"

Table 4.4: Studies selected from start set.

Forward snowballing from the start set yielded 29 initial candidate papers. After applying selection criteria, 10 candidate papers remained, of which all satisfied the minimum study quality criteria. The papers selected from forward snowballing can be found in Table 4.6.

The total number of papers to review was thus 24.

4.4.2 Data Extraction

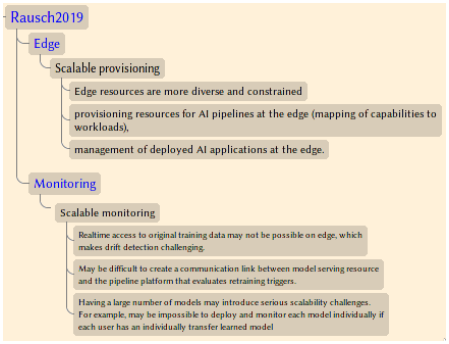
Data extraction proceeded by reading the literature and extracting information as described in Section 4.3.4 and entering the data in a spreadsheet.

Study	Title
Li *et al.* [35]	"Scaling Machine Learning as a Service"
Baylor *et al.* [36]	"TFX"
Crankshaw *et al.* [37]	"Clipper: A Low-Latency Online Prediction Serving System"
Lwakatare *et al.* [38]	"A Taxonomy of Software Engineering Challenges for Machine Learning Systems: An Empirical Investigation"
Bernardi *et al.* [39]	"150 Successful Machine Learning Models"
Garcia *et al.* [40]	"A Cloud-Based Framework for Machine Learning Workloads and Applications"

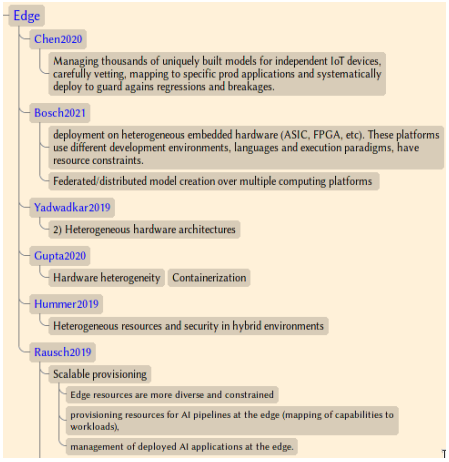
Table 4.5: Studies selected from backward snowballing.

Study	Title
Yadwadkar *et al.* [41]	"A Case for Managed and Model-less Inference Serving"
Rausch and Dustdar [42]	"Edge Intelligence: The Convergence of Humans, Things, and AI"
Rausch *et al.* [43]	"Towards a Serverless Platform for Edge AI"
Peticolas *et al.* [44]	"Mímir: Building and Deploying an ML Framework for Industrial IoT"
Chahal *et al.* [45]	"Migrating a Recommendation System to Cloud Using ML Workflow"
Zhang *et al.* [46]	"Model-Switching: Dealing with Fluctuating Workloads in Machine-Learning-as-a-Service Systems"
Pääkkönen *et al.* [47]	"Architecture for Enabling Edge Inference via Model Transfer from Cloud Domain in a Kubernetes Environment"
Gupta *et al.* [48]	"Containerized Architecture for Edge Computing in Smart Home : A consistent architecture for model deployment"
Richins *et al.* [49]	"Missing the Forest for the Trees: End-to-End AI Application Performance in Edge Data Centers"
Choi *et al.* [50]	"Lazy Batching: An SLA-aware Batching System for Cloud Machine Learning Inference"

Table 4.6: Studies selected from forward snowballing.



(a) Example of author-oriented data using [42] as an example.



(b) Example of code-oriented data related to edge deployments.

Figure 4.1: Illustration of hierarchy swap in the coding process. In Figure 4.1a, the data is author-oriented after initial coding. After applying the hierarchy-swap feature, data is clustered by the *Edge* code in Figure 4.1b.

4.4.3 Data Synthesis

Data synthesis with thematic analysis was performed as described in Section 4.3.5 with the help of TreeSheets[3], a hierarchical spreadsheet application. For each research question, relevant extracted data for each author was entered in the hierarchy. The data pieces were then coded according to their content. Lastly, the hierarchy swap feature was used to reorganize the data from being author-oriented to being code-oriented, which enabled discovery of commonalities and differences between studies. An example of this is shown in Figure 4.1.

[3] `https://strlen.com/treesheets/`

Chapter 5

Research Results

This chapter presents the results from the SLR on a per-research question basis.

5.1 RQ1: How Are ML Models Operationalized in the State of the Art?

5.1.1 Packaging and Integration

Packaging models in containers is a commonly reported method of integration, in which the model is accessible through representational state transfer (REST) or remote procedure call (RPC) interfaces [33, 35, 37, 40]. When deploying numerous different models, using a consistent standard for application programming interfaces (APIs), such as OpenAPI[1], across models can facilitate system integration [40]. Another reported method of integration is to serialize the model (e.g. into ONNX[2], HDF5, Joblib[3]) and load it at runtime, possibly with an ML framework that is optimized for production [27, 44, 45, 47]. The model could also be packaged in a format specific to an end-to-end framework such as MLflow[4] [31]. The model may be integrated directly into the application code [30, 33, 34], traditionally first having to be rewritten for production [27].

5.1.2 Deployment

Deployment, the transitioning of a packaged and integrated model into a serving state, may occur through a few different methods. Models packaged in containers are simply run directly as standalone services [30, 33–35, 40]. However, models may be deployed in a target environment that is different from where they are packaged, in which case the model transfer may happen through either a push- or a pull-pattern.

[1] https://www.openapis.org/
[2] https://onnx.ai/
[3] https://joblib.readthedocs.io/en/latest/
[4] https://mlflow.org/

In a pull-pattern deployment, the target environment (host application running on e.g. server or edge device) periodically polls for model updates and downloads when available [35, 44, 47].

In a push-pattern deployment, the target environment is notified of the availability of a new model by a master server, e.g. by the server where the model was trained. This will happen either through a messaging service [30, 40], where message contains metadata including the location of the updated model, or by the model being pushed directly to the target environment through some receiving interface [47].

When an updated model has reached the target environment, subsequent redeployments can happen in a few different ways. For models packaged as standalone containers, a container orchestration service (e.g. Kubernetes[5]) can roll out updated models without downtime. Serialized models may simply be loaded into memory by the application, potentially leading to downtime. If the hosting application is containerized, model redeployment may be orchestrated by deploying a new application instance with the new model data, alongside the old instance. When the new instance is finished initializing, it can start serving requests, and the old instance may be evicted [47]. To avoid high response latencies for the first query (cold-start issues), models may be queried with an empty request (or an explicit warm-up function call may be used if available) which forces lazy-loaded model components to initialize [35].

When using an ML deployment service (e.g. SageMaker[6]), the service may handle the deployment of serialized models [45].

Models intended for batch predictions may simply be plugged into a computing pipeline such as Apache Spark[7], computing predictions and storing them in a database or data warehouse [35].

5.1.3 Serving and Inference

Models are commonly made available for serving predictions through a REST [29, 30, 33, 37, 40, 47] or RPC [33, 35, 37] API. To meet inference service level objectives (SLOs), there are several techniques reported. One is model-switching, where less accurate but more performant models are used during periods of high load in order to meet required SLOs [46]. Another is to use adaptive batching queues with a timeout, where queries are batched together in batch sizes that are tuned to the individual model and framework. During periods of low traffic, the timeout is reached before the query batch is filled, and inference is run on the batch in order to not exceed the SLO [37]. Additionally, a cache layer may be put on top of the model to reduce computation [37]. In order to avoid cold-start issues caused by model loading and initialization, models should be warmed up at deployment and be kept perpetually warm over the course of its life [46]. Model

[5] https://kubernetes.io/

[6] https://aws.amazon.com/sagemaker/

[7] https://spark.apache.org/

warmup can be achieved through either a method provided by the ML framework [35], or more generally through issuing an empty query against the model [40].

5.1.4 Monitoring and Logging

Runtime monitoring is important for operationalized ML in order to increase trust and detect performance degradations [35, 42]. When the model is operational, the whole application stack is constantly monitored for performance metrics such as latency, throughput, disk utilization, etc [33, 44]. Predictions are logged, and when available joined with actual outcomes[35]. Model accuracy should be used to determine when a new model is needed [44]. The application should be validated against predefined key performance indicators (KPIs) of the project [33].

5.2 RQ2: What Are the Main Challenges in Operationalizing ML Models?

5.2.1 Packaging and Integration

ML frameworks are not all created equal, and some may be better suited for research than production or vice versa, typically because of performance characteristics versus support for rapid experimentation and debugging. [37] notes several challenges in deploying ML frameworks: interface inconsistency across frameworks, changes over time in what is the best framework for the job, and frameworks that are not optimized for deployment. Developers then have to make the choice of either using a suboptimal framework, or incurring technical debt in order to support and integrate multiple frameworks.

[27] reports solving this tradeoff by decoupling the frameworks from the model by transferring it from research to production using an exchange format for neural networks. [37] also uses decoupling as a solution to this problem, but opts for adding an abstraction layer with a simple interface on top of all models. This solution is more general, as it is not specific to neural networks.

[32] reports the extensive use of glue code as a challenge with which companies struggle. Glue code is code that merely glues together different parts of the program, without itself providing any functionality. The authors also highlight the integration of data-driven models with computation-driven software components as a potentially non-trivial task.

5.2.2 Serving

Serving-related challenges are some of the most often reported ones in the literature [32, 35–41, 45, 46, 48–50], usually from a performance perspective.

Achieving a low inference latency and high throughput is widely reported as a challenge in serving ML models [32, 35, 37–39]. Bernardi *et al.* [39] found that there is a statistically significant negative correlation between latency and

conversion rate at Booking.com, highlighting the business impact of serving performance.

Several potential solutions for improving serving performance have been proposed in the literature. Gupta *et al.* [48] suggests deploying ML models to the edge in order to reduce network latency. However, edge deployment bring a whole host of its own challenges, which will be discussed in Section 5.2.3.

In a system utilizing publish/subscribe messaging services, [49] observes that 33% of application latency is a result of internal communication frameworks, and by experimentation find that the messaging system becomes congested when faced with an eightfold increase in inference performance. With a growing number of specialized ML accelerator hardware, it not inconceivable for such performance bottlenecks to appear in the near future. The authors find that a solution to the problem could be to increase the number of broker node instances used in the messaging system, which clears up the congestion issue. However, ML models typically do not make inferences based on raw data, but may require data transformations both before and after inference. [49] warns that the data processing code in streaming ML systems could soon become another performance bottleneck. The authors do not, however, offer a solution.

Start-up/cold-start latency is a typical serving challenge for ML models [41]. It is mainly caused by the ML model being loaded into memory and initialized, and is typically solved by warming up the model after it has been deployed, and then keeping it perpetually warm [46]. There are, however, problems with this solution, which will be discussed in Section 5.4.

[46] discusses the problem of meeting performance SLOs during significant variations in model query traffic, a problem also observed by [41]. The authors note that currently, the options are to either accept degraded throughput or latency during high-demand scenarios, or to scale up the hardware resources (as suggested by [39]) and thereby incurring increased costs. The difficulty in achieving cost-effective and performant inference is also corroborated by [45], where the authors observe that increasing hardware instances and network bandwidth both improve inference latencies. To solve the issue of upholding latency SLOs while minimizing costs, the model-switching technique is proposed by [46] as a solution, whereby high-accuracy/low-performance models are traded for lower-accuracy/high-performance models during load spikes in order to meet effective accuracy SLO, i.e. the average accuracy of predictions that meet latency requirements.

5.2.3 Edge

Edge deployments are often reported as being more challenging than cloud deployments because of resource heterogeneity and the typically distributed nature of edge devices [28, 31, 32, 41, 48]. Architecture and capabilities of the hardware, varying network connection constraints

Reviewed studies reporting deployment of ML models to Internet of Things

(IoT) devices tend to construct custom deployment systems in order to deploy to edge devices [44].

5.2.4 Monitoring and Logging

Monitoring ML models is challenging [35, 39], but is required in order to detect performance degradations and build confidence in the model [31, 32]. After making an inference, the true label may not be available for an extended period of time, making it difficult to monitor prediction quality [39]. One solution used in practice is to look at the distribution of predictions and determine if there is a significant deviation from what the response distribution of an ideal model would look like [39].

5.3 RQ3: What Tools and Software Infrastructure Are Used to Operationalize ML Models?

In the space of ML-specific platforms, there are a wide array of options reviewed by [33] covering all or parts of the project and model lifecycle. Polyaxon[8], MLflow[9], TFX[10], ZenML[11], Flyte[12], Seldon Core[13] and BentoML[14] are some of the more fully-integrated platforms that exist [31, 33], as well as the Amazon-exclusive SageMaker[15] [33, 45]. For the sake of brevity, and to avoid simply reiterating all tools mentioned in [33], the reader is instead referred to their paper for an overview of various tools for operationalizing ML.

Containerization is a commonly reported method of packaging ML models [33, 42, 44], and Docker[16] is by far the most frequently reported solution [29, 35, 37, 40, 47–49]. Models packaged in containers are typically accessed through either a REST API [40, 47], such as Flask[17] [48], or RPC interface [33, 37], such as Apache Thrift[18] [35]. Having multiple containers deployed will often necessitate the use of a container orchestration tool such as Kubernetes[19] [30, 43, 47, 49] or Apache Mesos[20] [40]. However, when deploying on edge clusters, K3s[21] may be preferrable [47]. Service for Kubernetes and K3s may be defined with service

[8] https://polyaxon.com/
[9] https://mlflow.org/
[10] https://www.tensorflow.org/tfx/
[11] https://zenml.io/
[12] https://flyte.org/
[13] https://www.seldon.io/tech/products/core/
[14] https://www.bentoml.ai/
[15] https://aws.amazon.com/sagemaker/
[16] https://www.docker.com/
[17] https://flask.palletsprojects.com/en/2.0.x/
[18] https://thrift.apache.org/
[19] https://kubernetes.io
[20] https://mesos.apache.org/
[21] https://k3s.io/

descriptors like Helm[22] [47]. In order to manage multiple clusters of containers, a Kubernetes-as-a-service tool such as Rancher[23] may be used [47].

Serverless frameworks like Apache OpenWhisk[24] and Knative[25] may be used to deploy models as functions and provide horizontal scaling [40, 43], both building on top of Kubernetes.

When models are downloaded by the target deployment environment, a static storage service such as AWS S3[26] may be used for model storage [30, 45].

A commonly reported messaging and communications framework is Apache Kafka[27], used for logging predictions [35], notify model hosts of updates [30, 40] and general inter-container communication [49].

A large proportion of infrastructure and tooling for ML are Kubernetes-based, reflecting the active ecosystem around the popular container orchestration tool. This could be a barrier to MLOps adoption, as Ruf *et al.* [33] notes that setting up a Kubernetes cluster is an expensive task.

5.4 RQ4: What Are the Gaps in Current Tooling and Infrastructure Used to Operationalize ML Models?

Many opportunities for further work and research have been identified in the reviewed literature. In particular, [41] and [46] identify a large number of research directions in the area of model serving performance. [32], [42] and [48] also propose several areas of research for edge ML deployment.

5.4.1 Model-switching

In the context of an ML system with multiple available variants of the same model (with different accuracy and performance characteristics) and multiple hardware resources to choose from, [41] presents a series of interrelated open problems concerned with serving performance. The automatic selection of model variant based on a given SLO will be referred to as *model-switching* based on the terminology used in [46]. The overarching question is that of how the serving system should automatically select a model variant and underlying hardware given some SLO specified by a client application.

First, given an inference query, how should it be placed/scheduled by the serving system? The options are to either generate a new model variant, to load an existing model variant, or to query an already loaded model. The choice will depend on the individual application requirements, e.g. whether the inference is online or offline, what the SLOs are, etc. [41] and [46] propose that further research

[22] https://helm.sh/
[23] https://rancher.com/
[24] https://openwhisk.apache.org/
[25] https://knative.dev
[26] https://aws.amazon.com/s3/
[27] https://kafka.apache.org/

be conducted into how to synergistically combine model-switching and hardware resource selection.

Further, given the choice of placement, a new model may need to be loaded, additional hardware resources may need to be launched, or loaded models may need to be evicted to free up resources. As [46] points out, keeping all models warm at all times is prohibitively expensive and scales poorly. The model management operations add latency to the query response, affecting the system's ability to meet SLOs. [41] and [46] encourage research into methods for reducing or avoiding this latency, striving for the performance of perpetually warm models, while aiming for the resource usage of cold models.

Next, the question is how to capture and organize the data needed for making a decision on query placement. In addition to the set of possible hardware resources and model variants, the serving system must also take into account the current state of the system, which is continually changing. The current state of the system is described by which model variants are loaded on which hardware resources, which model variants exist but are not loaded, which model variants do not exist but are able to be generated, CPU/memory/disk/network usage of each hardware instance, inference query traffic. All of this data needs to be captured and organized in a way that enables fast and efficient decision making on query placement.

Not only does the logical aspect of hardware and model variant selection need to be considered, the geographical location of hardware resources must also be taken into account. Given that resources may lie anywhere on the core-edge continuum, [41] requests research into answering the questions of where models should be loaded, where resource management decisions should be made, and where query placement decisions should be made. [43] points out that edge resource management from the cloud is complex, partly because nodes may be on private networks or behind firewalls. In addition, edge resources may have unreliable or limited network access, making it challenging to maintain constant communication channels. The authors report that no out-of-the-box solution currently solves these issues, and request research into transparent edge-cloud resource consolidation/orchestration without the use of point-to-point integrations like VPNs.

[46] further proposes research into combining model-switching with performance optimization techniques like query caching and batching. Several batching techniques have been reported for ML serving systems. [37] uses a batching technique with a static maximum batch size with a timeout window which are both tuned on a per-model basis, while [50] proposes an adaptive batching scheme specifically for neural networks, reducing latency during low-traffic scenarios. Combining caching and batching with model-switching is an open problem for further work.

[46] proposes research into extending model-switching to additional types of computing resources. central processing units (CPUs) are currently the most widely used for inference in existing machine learning-as-a-service (MLaaS) plat-

forms, other types of hardware exist that are suited for neural network inference, such as graphics processing units (GPUs), tensor processing units (TPUs), etc. Research into model-switching which takes into account these additional types of hardware resources should be conducted to identify possible performance benefits. Further, the additional hardware heterogeneity could pose an interesting challenge when combining model-switching and hardware selection during dynamic query placement.

5.4.2 Edge Deployment

[43] observes that the performance of the Kubernetes scheduler begins degrading with 5000 nodes and fewer than 10 constraints, struggling to process more than 15 functions per second. It is conceivable that the number of edge devices may reach the thousands and beyond in an IoT scenario, e.g. in industry 4.0, suggesting that the scheduler may become a performance bottleneck in the future. The authors therefore propose research into more scalable function scheduling.

[32] and [42] report a lack of generic solutions for deploying ML models to embedded and edge devices. [42] reports that conventional IoT provisioning frameworks are not suited for deploying ML models, while [48] reports that there is a lack of solutions for deploying to edge devices that do not support containerization techniques. [43] reports that support for non-x86 architectures is lacking.

Edge devices present greater heterogeneity in computing capabilities, storage and networking, and in the context of having multiple models deployed in an edge architecture, some models may have to be evicted from time to time in order to free up resources for new deployments. [42] and [43] propose research into strategies for intelligently evicting models from edge devices.

According to [42], monitoring edge-deployed ML models can potentially present two different challenges. First, some metrics, such as concept drift and model drift, require continuous access to the original training set. However, edge devices may not have access to the training set for multiple reasons, for example because of storage space constraints or because of data privacy concerns. Further research is required to solve the problem of drift detection in edge devices where training data is unavailable.

Secondly, [42] identifies the problem of triggering centralized model retraining. Retraining triggers require that monitoring data be continuously streamed from the edge devices to the (possibly centralized) ML pipeline. Not only could this be difficult because of unreliable network connections, bandwidth restrictions or data usage constraints; handling large volumes of incoming monitoring data from edge devices may itself present a challenge of scalability.

On the topic of monitoring, [31] highlights a need for framework-agnostic model telemetry solutions to enable the capture of statistical performance metrics with a broad set of supported deployment environments.

[31] also notes the need for a general representation format for multistep ML workflows. Many processes in ML pipelines contain multiple subtasks, where

changing a single step may cause regressions. A new format should, according to the authors, provide explicit IO interfaces and enable parallel execution of independent subtasks with existing workflow execution systems (e.g. Airflow[28]).

[28] `https://airflow.apache.org/`

Chapter 6

Discussion

Section 6.1 will discuss how this study's results compare with those of previous related work. Section 6.2 will discuss limitations of this study, as well as steps taken to mitigate some of the threats to validity.

6.1 Comparison to Related Work

This study has contributed the following new knowledge.

- How ML models are operationalized with respect to tools and infrastructure. This includes various techniques for packaging and integration, deployment, serving and monitoring, which to the author's best knowledge have not been investigated in any previous SLR.
- An overview of what tools have reportedly been used to operationalize ML models in the literature.
- An overview of areas for further research into ML operationalization as suggested by the literature.

The results of the review show that researchers should work on two areas. First, performance and scalability in prediction serving, possibly using a combination of existing techniques for batching, caching and dynamic model selection. Second, edge environments in general were reported to have general open problems in deployment and monitoring.

The results of the study could be of value to industry practitioners as a reference for various techniques and patterns for operationalizing ML. In addition, the study could also serve as a starting point for understanding what tools are used in different stages and contexts of operationalization, as well as what challenges may be expected when operationalizing ML models.

6.2 Limitations

6.2.1 Threats to Internal Validity

Being an SLR, there are certain inherent threats to validity. There is the possibility that not all relevant literature is found or included. To mitigate this, the start set included papers spanning multiple years (2017-2021), venues (IEEE, ACM, Sciendo, Frontiers, Sciencedirect, Zenodo, IGI Global, Wiley, MDPI, Springer) and authors. However, as the snowballing procedure was only limited to a single iteration of forward and backward snowballing, instead of continuing until no new studies are found, it is quite probable that this review is not exhaustive.

As the review was conducted by a single (and inexperienced) person, there is the probability of bias and mistakes in all stages of the review process. The application of selection criteria and quality assessment criteria may have been inconsistent or biased, information may have been overlooked during the data extraction process, and the analysis could contain unfounded or biased conclusions. In particular, as a simplified thematic analysis procedure was followed, the data synthesis was not as rigorous as suggested by [26]. Furthermore, qualitative studies in general are more closely tied to the researchers' background, identity, assumptions and beliefs than quantitative studies according to [51]. To counteract some of these personal biases, possible issues were discussed with the supervisor. A more robust approach, however, would have had at least one other researcher conducting each step of the review in parallel, using tests such as Fleiss' Kappa to determine level of aggreement between researchers. In addition, a sensitivity analysis could have been performed by synthesizing data from a random subset of the reviewed literature to determine if the conclusion would differ significantly.

6.2.2 Threats to External Validity

This study suffers from some unmitigated threats to external validity, factors which may render the conclusions of a study inapplicable outside the context of the study. As the field of MLOps is still young and rapidly evolving, there is the strong possibility of information reported in published research being outdated shortly after publication. This review includes studies published as early as 2017, meaning that the conclusions drawn may be based on outdated data and therefore be of limited value. To mitigate this, the selection criteria for recency could have been made stricter, e.g. only including studies published in 2020 and later. However, it is not known if any of the reviewed studies are outdated or if the quality of this review would have improved by excluding some of the older papers.

Additionally, the study potentially misses out on practitioner knowledge because GL is not included in the review. While it is more challenging to assess the quality and reliability of GL, it could provide valuable practitioner insights, especially in a practitioner-oriented field like SE [25]. A GLR was not conducted in this study because of time constraints, but would be a natural direction to follow in future research.

Chapter 7

Future Work

Several opportunities for future research were identified in Section 5.4. The most prominent opportunities identified are related to serving performance and edge ML operationalization. In serving, there is the open problem of optimizing query response times with respect to accuracy and resource scaling costs during load spikes. In edge deployments, there are an array of opportunities for further research, related to scalable monitoring, scalable scheduling, and scalable provisioning of heterogeneous hardware resources.

Although initially intended to be an MLR, the scope of this study had to be reduced to only encompass an SLR because of time constraints. A natural next step would thus be to conduct a GLR to investigate practitioner knowledge on the subject using a similar methodology and research questions.

ML model management, also referred to as ModelOps [28], is a closely related field concerned with model lifecycle management, tracking model versions and model lineage. There exists no published literature reviews on the subject, despite being an important component for ensuring proper AI governance. Further research on the topic should therefore be conducted in the future.

Bibliography

[1] P. Bourque, *SWEBOK : guide to the software engineering body of knowledge*. Los Alamitos, CA: IEEE Computer Society, 2014, ISBN: 0769551661.

[2] F. Kumeno, "Sofware engneering challenges for machine learning applications: A literature review," *Intelligent Decision Technologies*, vol. 13, pp. 463–476, 2020, ISSN: 18724981, 18758843. DOI: 10.3233/IDT-190160.

[3] E. Nascimento, A. Nguyen-Duc, I. Sundbø, and T. Conte, "Software engineering for artificial intelligence and machine learning software: A systematic literature review," Nov. 2020. arXiv: 2011.03751 [cs.SE].

[4] B. Kitchenham and S. Charters, *Guidelines for performing systematic literature reviews in software engineering*, 2007.

[5] 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 - EASE '14*, ACM Press, 2014. DOI: 10.1145/2601248.2601268.

[6] "The state of AI in 2020," 2020. [Online]. Available: https://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/global-survey-the-state-of-ai-in-2020.

[7] S. Schlögl, C. Postulka, R. Bernsteiner, and C. Ploder, "Artificial intelligence tool penetration in business: Adoption, challenges and fears," in, Springer International Publishing, 2019, pp. 259–270. DOI: 10.1007/978-3-030-21451-7_22.

[8] G. Giray, "A software engineering perspective on engineering machine learning systems: State of the art and challenges," *Journal of Systems and Software*, vol. 180, p. 111 031, Oct. 2021. DOI: 10.1016/j.jss.2021.111031.

[9] S. Martínez-Fernández, J. Bogner, X. Franch, M. Oriol, J. Siebert, A. Trendowicz, A. M. Vollmer, and S. Wagner, "Software engineering for ai-based systems: A survey," May 2021. arXiv: 2105.01984 [cs.SE].

[10] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, 2015.

[11] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "DevOps," vol. 33, no. 3, pp. 94–100, May 2016. DOI: 10.1109/ms.2016.68.

[12] B. B. N. de França, H. Jeronimo, and G. H. Travassos, "Characterizing DevOps by hearing multiple voices," ACM Press, 2016. DOI: 10.1145/2973839.2973845.

[13] J. Soh and P. Singh, "Machine learning operations," in *Data Science Solutions on Azure*, Apress, 2020, pp. 259–279. DOI: 10.1007/978-1-4842-6405-8_8.

[14] M. Shahin, M. A. Babar, and L. Zhu, "Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices," *IEEE Access*, vol. 5, pp. 3909–3943, 2017. DOI: 10.1109/access.2017.2685629.

[15] P. Rodríguez, A. Haghighatkhah, L. E. Lwakatare, S. Teppola, T. Suomalainen, J. Eskeli, T. Karvonen, P. Kuvaja, J. M. Verner, and M. Oivo, "Continuous deployment of software intensive products and services: A systematic mapping study," *Journal of Systems and Software*, vol. 123, pp. 263–291, Jan. 2017. DOI: 10.1016/j.jss.2015.12.015.

[16] L. Baier, F. Jöhren, and S. Seebacher, "Challenges in the deployment and operation of machine learning in practice," in *ECIS 2019 - 27th European Conference on Information Systems*, May 2019. [Online]. Available: https://www.researchgate.net/publication/332996647_CHALLENGES_IN_THE_DEPLOYMENT_AND_OPERATION_OF_MACHINE_LEARNING_IN_PRACTICE.

[17] A. Paleyes, R.-G. Urma, and N. D. Lawrence, "Challenges in deploying machine learning: A survey of case studies," *The ML-Retrospectives, Surveys & Meta-Analyses Workshop, NeurIPS 2020*, arXiv:2011.09926, Nov. 2020. arXiv: 2011.09926 [cs.LG]. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2020arXiv201109926P.

[18] L. E. Lwakatare, A. Raj, I. Crnkovic, J. Bosch, and H. H. Olsson, "Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions," *Information and Software Technology*, vol. 127, p. 106 368, Nov. 2020. DOI: 10.1016/j.infsof.2020.106368.

[19] L. E. Lwakatare, I. Crnkovic, E. Rånge, and J. Bosch, "From a data science driven process to a continuous delivery process for machine learning systems," in *Product-Focused Software Process Improvement*, Springer International Publishing, 2020, pp. 185–201. DOI: 10.1007/978-3-030-64148-1_12.

[20] 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 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, ACM, Oct. 2020. DOI: 10.1145/3382494.3410681.

[21] M. M. John, H. H. Olsson, and J. Bosch, "Architecting ai deployment: A systematic review of state-of-the-art and state-of-practice literature," in *Lecture Notes in Business Information Processing*, Springer International Publishing, 2021, pp. 14–29. DOI: 10.1007/978-3-030-67292-8_2.

[22] G. Lorenzoni, P. Alencar, N. Nascimento, and D. Cowan, "Machine learning model development from a software engineering perspective: A systematic literature review," Feb. 2021. arXiv: 2102.07574 [cs.SE].

[23] A. Serban and J. Visser, "An empirical study of software architecture for machine learning," May 2021. arXiv: 2105.12422 [cs.SE].

[24] M. M. John, H. H. Olsson, and J. Bosch, "Towards MLOps: A framework and maturity model," IEEE, Sep. 2021. DOI: 10.1109/seaa53835.2021.00050.

[25] V. Garousi, M. Felderer, and M. V. Mäntylä, "The need for multivocal literature reviews in software engineering," in *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, ACM, Jun. 2016. DOI: 10.1145/2915970.2916008.

[26] C. Lochmiller, "Conducting thematic analysis with qualitative data," *The Qualitative Report*, Jun. 2021. DOI: 10.46743/2160-3715/2021.5008.

[27] K. Hazelwood, S. Bird, D. Brooks, S. Chintala, U. Diril, D. Dzhulgakov, M. Fawzy, B. Jia, Y. Jia, A. Kalro, J. Law, K. Lee, J. Lu, P. Noordhuis, M. Smelyanskiy, L. Xiong, and X. Wang, "Applied machine learning at facebook: A datacenter infrastructure perspective," IEEE, Feb. 2018. DOI: 10.1109/hpca.2018.00059.

[28] W. Hummer, V. Muthusamy, T. Rausch, P. Dube, K. E. Maghraoui, A. Murthi, and P. Oum, "ModelOps: Cloud-based lifecycle management for reliable and trusted AI," IEEE, Jun. 2019. DOI: 10.1109/ic2e.2019.00025.

[29] R. Krishnamurthi, R. Maheshwari, and R. Gulati, "Deploying deep learning models via IOT deployment tools," IEEE, Aug. 2019. DOI: 10.1109/ic3.2019.8844946.

[30] Y. Liu, Z. Ling, B. Huo, B. Wang, T. Chen, and E. Mouine, "Building a platform for machine learning operations from open source frameworks," vol. 53, no. 5, pp. 704–709, 2020. DOI: 10.1016/j.ifacol.2021.04.161.

[31] A. Chen, A. Chow, A. Davidson, A. DCunha, A. Ghodsi, S. A. Hong, A. Konwinski, C. Mewald, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, A. Singh, F. Xie, M. Zaharia, R. Zang, J. Zheng, and C. Zumar, "Developments in MLflow, A system to accelerate the machine learning lifecycle," in *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*, ACM, Jun. 2020. DOI: 10.1145/3399579.3399867.

[32] J. Bosch, H. H. Olsson, and I. Crnkovic, "Engineering AI systems," in, IGI Global, 2021, pp. 1–19. DOI: 10.4018/978-1-7998-5101-1.ch001.

[33] P. Ruf, M. Madan, C. Reich, and D. Ould-Abdeslam, "Demystifying MLOps and presenting a recipe for the selection of open-source tools," vol. 11, no. 19, p. 8861, Sep. 2021. DOI: 10.3390/app11198861.

[34] T. Granlund, V. Stirbu, and T. Mikkonen, "Towards regulatory-compliant MLOps: Oravizio's journey from a machine learning experiment to a deployed certified medical product," vol. 2, no. 5, Jun. 2021. DOI: 10.1007/s42979-021-00726-1.

[35] L. E. Li, E. Chen, J. Hermann, P. Zhang, and L. Wang, "Scaling machine learning as a service," in *Proceedings of The 3rd International Conference on Predictive Applications and APIs*, C. Hardgrove, L. Dorard, K. Thompson, and F. Douetteau, Eds., ser. Proceedings of Machine Learning Research, vol. 67, PMLR, Nov. 2017, pp. 14–29. [Online]. Available: https://proceedings.mlr.press/v67/li17a.html.

[36] D. Baylor, E. Breck, H.-T. Cheng, N. Fiedel, C. Y. Foo, Z. Haque, S. Haykal, M. Ispir, V. Jain, L. Koc, C. Y. Koo, L. Lew, C. Mewald, A. N. Modi, N. Polyzotis, S. Ramesh, S. Roy, S. E. Whang, M. Wicke, J. Wilkiewicz, X. Zhang, and M. Zinkevich, "TFX," ACM, Aug. 2017. DOI: 10.1145/3097983.3098021.

[37] D. Crankshaw, X. Wang, G. Zhou, M. J. Franklin, J. E. Gonzalez, and I. Stoica, "Clipper: A low-latency online prediction serving system," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, Boston, MA: USENIX Association, Mar. 2017, pp. 613–627, ISBN: 978-1-931971-37-9. [Online]. Available: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/crankshaw.

[38] L. E. Lwakatare, A. Raj, J. Bosch, H. H. Olsson, and I. Crnkovic, "A taxonomy of software engineering challenges for machine learning systems: An empirical investigation," in *Lecture Notes in Business Information Processing*, Springer International Publishing, 2019, pp. 227–243. DOI: 10.1007/978-3-030-19034-7_14.

[39] L. Bernardi, T. Mavridis, and P. Estevez, "150 successful machine learning models," ACM, Jul. 2019. DOI: 10.1145/3292500.3330744.

[40] A. L. Garcia, J. M. D. Lucas, M. Antonacci, W. Z. Castell, M. David, M. Hardt, L. L. Iglesias, G. Molto, M. Plociennik, V. Tran, A. S. Alic, M. Caballer, I. C. Plasencia, A. Costantini, S. Dlugolinsky, D. C. Duma, G. Donvito, J. Gomes, I. H. Cacha, K. Ito, V. Y. Kozlov, G. Nguyen, P. O. Fernandez, Z. Sustr, and P. Wolniewicz, "A cloud-based framework for machine learning workloads and applications," vol. 8, pp. 18 681–18 692, 2020. DOI: 10.1109/access.2020.2964386.

[41] N. J. Yadwadkar, F. Romero, Q. Li, and C. Kozyrakis, "A case for managed and model-less inference serving," ACM, May 2019. DOI: 10.1145/3317550.3321443.

[42] T. Rausch and S. Dustdar, "Edge intelligence: The convergence of humans, things, and AI," IEEE, Jun. 2019. DOI: 10.1109/ic2e.2019.00022.

[43] T. Rausch, W. Hummer, V. Muthusamy, A. Rashed, and S. Dustdar, "Towards a serverless platform for edge AI," in *2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 19)*, Renton, WA: USENIX Association, Jul. 2019. [Online]. Available: https://www.usenix.org/conference/hotedge19/presentation/rausch.

[44] D. Peticolas, R. Kirmayer, and D. Turaga, "Mímir: Building and deploying an ML framework for industrial IoT," IEEE, Nov. 2019. DOI: 10.1109/icdmw.2019.00065.

[45] D. Chahal, R. Ojha, S. R. Choudhury, and M. Nambiar, "Migrating a recommendation system to cloud using ML workflow," ACM, Apr. 2020. DOI: 10.1145/3375555.3384423.

[46] J. Zhang, S. Elnikety, S. Zarar, A. Gupta, and S. Garg, "Model-switching: Dealing with fluctuating workloads in machine-learning-as-a-service systems," in *12th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 20)*, USENIX Association, Jul. 2020. [Online]. Available: https://www.usenix.org/conference/hotcloud20/presentation/zhang.

[47] P. Pääkkönen, D. Pakkala, J. Kiljander, and R. Sarala, "Architecture for enabling edge inference via model transfer from cloud domain in a kubernetes environment," vol. 13, no. 1, p. 5, Dec. 2020. DOI: 10.3390/fi13010005.

[48] N. Gupta, K. Anantharaj, and K. Subramani, "Containerized architecture for edge computing in smart home : A consistent architecture for model deployment," IEEE, Jan. 2020. DOI: 10.1109/iccci48352.2020.9104073.

[49] D. Richins, D. Doshi, M. Blackmore, A. T. Nair, N. Pathapati, A. Patel, B. Daguman, D. Dobrijalowski, R. Illikkal, K. Long, D. Zimmerman, and V. J. Reddi, "Missing the forest for the trees: End-to-end AI application performance in edge data centers," IEEE, Feb. 2020. DOI: 10.1109/hpca47549.2020.00049.

[50] Y. Choi, Y. Kim, and M. Rhu, "Lazy batching: An SLA-aware batching system for cloud machine learning inference," IEEE, Feb. 2021. DOI: 10.1109/hpca51647.2021.00049.

[51] B. J. Oates, *Researching Information Systems and Computing*. Sage Publications, Nov. 2005, 360 pp., ISBN: 1412902231.

[52] D. C. Liu, S. Rogers, R. Shiau, D. Kislyuk, K. C. Ma, Z. Zhong, J. Liu, and Y. Jing, "Related pins at pinterest," ACM Press, 2017. DOI: 10.1145/3041021.3054202.

[53] M. Haldar, M. Abdool, P. Ramanathan, T. Xu, S. Yang, H. Duan, Q. Zhang, N. Barrow-Williams, B. C. Turnbull, B. M. Collins, and T. Legrand, "Applying deep learning to airbnb search," ACM, Jul. 2019. DOI: 10.1145/3292500.3330658.

[54] R. Ciucu, F. Adochiei, I.-R. Adochiei, F. Argatu, G. Seriţan, B. Enache, S. Grigorescu, and V. V. Argatu, "Innovative devops for artificial intelligence," vol. 19, no. 1, pp. 58–63, Apr. 2019. DOI: 10.1515/sbeef-2019-0011.

[55] S. Jackson, M. Yaqub, and C.-X. Li, "The agile deployment of machine learning models in healthcare," vol. 1, Jan. 2019. DOI: 10.3389/fdata.2018.00007.

[56] B. Karlaš, M. Interlandi, C. Renggli, W. Wu, C. Zhang, D. M. I. Babu, J. Edwards, C. Lauren, A. Xu, and M. Weimer, "Building continuous integration services for machine learning," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, Jul. 2020. DOI: 10.1145/3394486.3403290.

[57] W.-J. van den Heuvel and D. A. Tamburri, "Model-driven ML-ops for intelligent enterprise applications: Vision, approaches and challenges," in, Springer International Publishing, 2020, pp. 169–181. DOI: 10.1007/978-3-030-52306-0_11.

[58] Y. Zhou, Y. Yu, and B. Ding, "Towards MLOps: A case study of ML pipeline platform," IEEE, Oct. 2020. DOI: 10.1109/icaice51518.2020.00102.

[59] D. A. Tamburri, "Sustainable MLOps: Trends and challenges," IEEE, Sep. 2020. DOI: 10.1109/synasc51798.2020.00015.

[60] B. M. A. Matsui and D. H. Goya, "Application of devops in the improvement of machine learning processes," en, 2020. DOI: 10.5281/ZENODO.4318113.

[61] S. Mariani, M. M. H. Lahr, E. Metting, E. Vargiu, and F. Zambonelli, "Developing an ML pipeline for asthma and COPD: The case of a dutch primary care service," *International Journal of Intelligent Systems*, Jul. 2021. DOI: 10.1002/int.22568.

Appendix A

Snowballing Start Set

Study	Title
Liu *et al.* [52]	"Related Pins at Pinterest"
Hazelwood *et al.* [27]	"Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective"
Haldar *et al.* [53]	"Applying Deep Learning to Airbnb Search"
Hummer *et al.* [28]	"ModelOps: Cloud-Based Lifecycle Management for Reliable and Trusted AI"
Ciucu *et al.* [54]	"Innovative Devops for Artificial Intelligence"
Krishnamurthi *et al.* [29]	"Deploying Deep Learning Models via IOT Deployment Tools"
Jackson *et al.* [55]	"The Agile Deployment of Machine Learning Models in Healthcare"
Karlaš *et al.* [56]	"Building Continuous Integration Services for Machine Learning"
Heuvel and Tamburri [57]	"Model-Driven ML-Ops for Intelligent Enterprise Applications: Vision, Approaches and Challenges"
Zhou *et al.* [58]	"Towards MLOps: A Case Study of ML Pipeline Platform"
Tamburri [59]	"Sustainable MLOps: Trends and Challenges"
Liu *et al.* [30]	"Building A Platform for Machine Learning Operations from Open Source Frameworks"
Matsui and Goya [60]	"Application of DevOps in the improvement of machine learning processes"
Chen *et al.* [31]	"Developments in MLflow"
Bosch *et al.* [32]	"Engineering AI Systems"
Martínez-Fernández *et al.* [9]	"Software Engineering for AI-Based Systems: A Survey"
Mariani *et al.* [61]	"Developing an ML pipeline for asthma and COPD: The case of a Dutch primary care service"
Ruf *et al.* [33]	"Demystifying MLOps and Presenting a Recipe for the Selection of Open-Source Tools"
Granlund *et al.* [34]	"Towards Regulatory-Compliant MLOps: Oravizio's Journey from a Machine Learning Experiment to a Deployed Certified Medical Product"

Table A.1: Studies in the snowballing start set.