A Multivocal Literature Review of MLOps Tools and Features

Gilberto Recupito*§, Fabiano Pecorelli*, Gemma Catolino†,
Sergio Moreschini*, Dario Di Nucci§, Fabio Palomba§, and Damian A. Tamburri‡,
*Tampere University, Tampere, Finland
†Tilburg University - JADS, 's-Hertogenbosch, The Netherlands
‡Eindhoven Technical University - JADS, 's-Hertogenbosch, The Netherlands
§SeSa Lab - University of Salerno, Fisciano, Italy
gilberto.recupito@tuni.fi, fabiano.pecorelli@tuni.fi, g.catolino@tilburguniversity.edu, sergio.moreschini@tuni.fi,
ddinucci@unisa.it, fpalomba@unisa.it, d.a.tamburri@tue.nl

***Abstract*—DevOps has become increasingly widespread, with companies employing its methods in different fields. In this context, MLOps automates Machine Learning pipelines by applying DevOps practices. Considering the high number of tools available and the high interest of the practitioners to be supported by tools to automate the steps of Machine Learning pipelines, little is known concerning MLOps tools and their functionalities. To this aim, we conducted a Multivocal Literature Review (MLR) to (i) extract tools that allow for and support the creation of MLOps pipelines and (ii) analyze their main characteristics and features to provide a comprehensive overview of their value. Overall, we investigate the functionalities of 13 MLOps Tools. Our results show that most MLOps Tools support the same features but apply different approaches that can bring different advantages, depending on user requirements.**

***Index Terms*—MLOps, Multivocal Literature Review, DevOps, Machine Learning**

I. Introduction

Effectively deploying new software artifacts as fast as possible is one of the most critical challenges for companies in the IT market [1]. To shorten software time to value, organizations apply DevOps, a model that foresees a strong collaboration between software development and operations teams. In this context, DevOps engineers develop reusable scripts to create, update and execute infrastructure through Infrastructure as Code (IaC) [2]. IaC provides many tools to deploy and operate cloud applications continuously. However, managing and configuring resources remains tricky and time-consuming, making IaC underutilized.

These issues become even more challenging when dealing with Machine Learning (ML) models, which do not solely consist of the ML code but also data sources, model management, and model deployment while maintaining prediction accuracy. Developing and operationalizing ML models requires novel development processes across the entire ML lifecycle that include new skills [3]; therefore, Machine Learning Operations (MLOps) has been introduced. Similar to DevOps, MLOps strongly advocates automation while monitoring all steps of the ML pipeline to help enterprises optimize workflows and avoid implementation issues. Although several tools have been introduced in the past years, MLOps remains a relatively new field with limited literature and online resources.

This paper provides an overview, feature analysis, and comparison of the MLOps tools available to bridge the different understanding between practitioners and academics and review the related tools and framework. In particular, we performed a multivocal literature review (MLR) [4] leveraging white (i.e., academic) and grey literature (e.g., reports, blog posts, white papers, and official documentation), extracting MLOps tools and analyzing them considering different characteristics.

Our results report the state-of-the-practice MLOps tools, each providing a diverse set of functionalities that support practitioners during all MLOps phases. We have analyzed the MLOps tools to investigate how MLOps tools offer these services. Our findings highlight the scalability and elasticity of Cloud-based ML Platforms, the high support for building customized solutions proposed by Orchestration Platforms, and the portability of TensorFlow Extended (TFX). These features have varying levels of priority based on the user requirements for ML applications. We deem our work of interest to both researchers and practitioners. No previous work has comprehensively compared MLOps features and tools, making it hard to find the most suitable components to manage ML pipelines and exploit MLOps benefits.

Structure of the paper. Section II provides an overview on MLOps. Section III summarizes the related work. Section IV describes the research design and research questions of our multivocal literature review, while the results are presented in Section V and the threats to validity in Section VI. Finally, Section VII concludes the paper and reports future work.

Online appendix. For the sake of space limitations, we store all the material of our multivocal literature review, including the list of primary studies, in an online appendix[1].

II. Background

Although ML provides predictions after being trained on a static dataset [5], its pipelines are much more complex. They require significant infrastructure engineering in addition

[1]https://github.com/gilbertrec/MLR-MLOps-Tools-Features-Appendix

to code [6], including data extraction, data analysis, data transformation, model training, model validation, model serving, and model monitoring. MLOps is the practice to help model, develop, and operationalize ML lifecycle drawing on the DevOps principles and practices [7].

The model considers the organizations' collaboration culture, levels of automation, the ability for continuous improvement and rapid workflow, performance measurement, shared responsibility, and communication when assessing maturity [8]. MLOps practices should be shared with all the stakeholders: from data scientists and data engineers to product managers and business people. Leveraging MLOps and automated workflows, organizations can deploy and refresh models more effectively and efficiently than with manual approaches [8]. However, developing ML pipelines and optimizing MLOps requires appropriate workflow management systems [9]. Over the years, many companies have provided support to build, scale, and manage end-to-end ML pipeline visually, allowing non-expert users to benefit from the ML potential [7]. Similar to DevOps, MLOps strongly advocates automation and monitoring at all steps of the ML pipeline with modular, self-contained components that aim to retrain models automatically and can be shared by different setups. Components consist of input parameters, outputs, and the location of the container images, the latter encapsulating the executable code and the environment definition. A typical MLOps workflow is composed of the following steps:

1) DATA EXTRACTION to integrate relevant data from various data sources.
2) DATA ANALYSIS to understand data from the datasets.
3) DATA CLEANING, TRANSFORMATION, AND FEATURE ENGINEERING to split data into training, validation, and test sets having appropriate formats.
4) MODEL TRAINING to train ML models and save the best performing trained model starting from different algorithms and parameter settings.
5) MODEL VALIDATION to evaluate models' quality interactively on the test data and identify whether they satisfy the quality criteria based on performance metrics.
6) MODEL SERVING to deploy models in the target environments integrated with other software components.
7) MODEL MONITORING to detect model degradation through usage, input data, and performance analyses.

The number of automated steps indicates the maturity level of the process. *Continuous Training* pipelines, deployed through CI/CD, perform continuous training and prediction of ML models on new data (i.e., Steps 1 to 5). We consider as MLOps tools all tools enabling the above steps.

III. RELATED WORK

O'Leary and Uchida [10] observed over 100 professional ML practitioners migrating ML models to systematic pipelines to devise common problems and opportunities. Their analysis revealed three hints to smooth the process: (i) the environment for prototyping ML models should be the same as production; (ii) ML pipelines should provide a framework of predefined canonical operations units as components instead of free-form flexibility; (iii) interfaces between components should be made explicit and straightforward to increase the ease to use. Hummer et al. [11] proposed a cloud-based framework and platform for end-to-end development and lifecycle management of artificial intelligence (AI) applications. They discuss the challenges of building scalable AI operations and implementing a ModelOps prototype around the concepts of metadata versioning, AI domain abstractions, re-usable patterns, event-based pipelines, and seamless integration of lifecycle capabilities. Benjamin et al. [7] provided 13 criteria to apply DevOps concepts to ML pipelines and proposed a pipeline-based meta-model based on two use cases (Dev and Ops in action). They conclude that models are crucial to explaining the decisions of both Dev and Ops teams, improving change reactivity, reliability, and automation. Lastly, several initiatives and projects focus on integrating DevOps and MLOps tools. Baylor et al. [12] described continuous pipelines in the Tensorflow Extended (TFX) platform developed and deployed at Google. In addition, they discuss the lessons learned while developing and deploying such models from the Google perspective. Kakantousis et al. [13] demonstrated the use of Hopsworks, a data and ML platform, and described all the stages of the pipeline. Fursin and Essayana [14] introduced CodeReef, an open MLOps platform that enables automated deployment of ML models.

While DevOps and IaC have become very popular among software practitioners, the increasing need to automate software building of ML-enabled systems presents new challenges. Although MLOps is considered the solution, little is known about the features, implications, and characteristics of MLOps tools that support its implementation.

IV. THE MULTIVOCAL LITERATURE REVIEW

Our paper investigates the state of the practice of MLOps tools to benchmark the main features and possibly choose the most suitable tool for building ML pipelines. We answer the following main research question:

What tools and features allow developers to build ML-enabled software systems?

which has been detailed into two sub-questions:

- RQ_1. *What MLOps tools can be used to build ML pipelines for Continuous Deployment?*
- RQ_2. *What are the main features offered by MLOps tools?*

The former research question gives an overview of the state of the practice of the MLOps tools available to practitioners, while the latter aims to discern a list of features to choose the most suitable solution for building automated ML pipelines. To answer these questions, we performed a multivocal literature review [4] leveraging white (i.e., academic) and grey literature (e.g., reports, blog posts, and official documentation), whose process is depicted in Figure 1.

We relied on the three stages defined by Kitchenham et al. [15]: (i) elaborate the search string, (ii) apply the string on

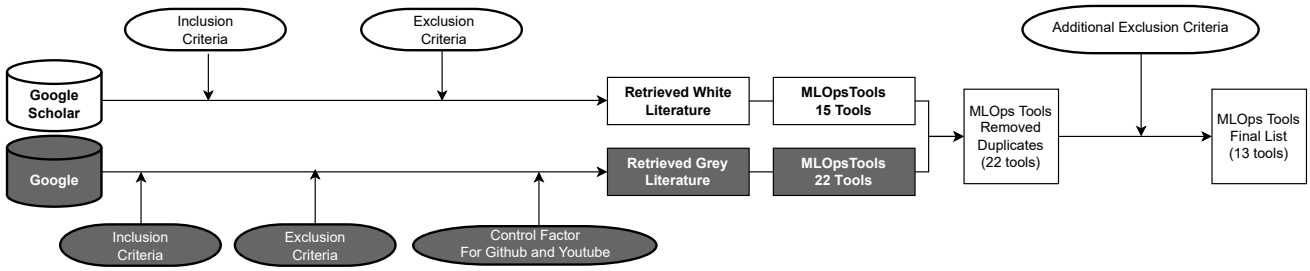

Fig. 1: Process of the Multivocal Literature Review.

chosen search engines, (iii) filter out and extract the studies based on inclusion and exclusion criteria.

The search string is based on the GQM terms to define the research goal by focusing on purpose, issue, object, and viewpoint [16].

Purpose: Systematically categorize
Issue: tools and applications for
Object (process): MLOps and DataOps
Viewpoint: from the practitioners' perspective.

The search query derived from the RQs is the following:

$$(mlops \vee machine\ learning\ ops \vee machine\ learning\ operation \vee dataops) \wedge (tool \vee application) \wedge (lifecycle \vee pipeline \vee platform \vee workflow)$$

We applied the search query to Google Scholar and Google. Google Scholar is used for the so-called "white" literature: books, journals, and thesis. The Google main search engine is needed to retrieve "grey" literature: blog posts, magazines, developer websites, webinars, GitHub repositories, and videos (YouTube) are used for our research. We used the incognito mode to avoid bias due to past search results and not avoided websites recommended by advertisements. The results were screened against inclusion and exclusion criteria. The inclusion criteria ensure that the focus is on the design, implementation, and experiences of MLOps tools or pipeline components. If a study meets one of the inclusion criteria, it is selected. The exclusion criteria ensure that studies adhere to design and implementation requirements. If a study meets one of the exclusion criteria, it is excluded. We defined our inclusion and exclusion criteria as follows:

Inclusion Criteria:

1) The study discusses the components of minimal end-to-end MLOps workflow(s).
2) The study discusses the practice of MLOps or ML-based applications.
3) The study refers to the implementation of MLOps tool(s).
4) The study discusses experience, opinions, or practices on MLOps pipeline(s).

Exclusion Criteria:

1) The study does not offer details concerning the design or implementation of MLOps tool(s).
2) The study solely offers the design of a specific component of ML pipeline(s).
3) The study does not offer or refer to details concerning ML automation.
4) The study refers to commercial platform(s) that offer(s) MLOps applications to sell their services for development and deployment.

To ensure that the retrieved literature have enough social relevance, we assigned additional control factors (i.e., popularity) for GitHub repositories and YouTube videos. A repository should have at least 100 stars indicating that practitioners endorse the project. The threshold for YouTube videos is set to 1000 views. The thresholds are based on observations of selected previous studies analyzing the two source types, which indicate that studies below these values are not appreciated. Two of the authors of this paper conducted the validation. They distributed the material nearly equally and validated only their corresponding instances. In problematic cases, the authors agreed on whether those specific documents should be considered. Whenever they disagreed, they involved one or more other authors of the paper in resolving the disagreement. Finally, to answer RQ2, we extracted the main features of the retrieved MLOps tools by analyzing their documentation. Please consider that we cover the ML pipeline, from data extraction to model monitoring. We analyzed the selected sources studies following a qualitative analysis process [17] by applying descriptive and pattern coding [18]. In detail, to obtain the tools, the features and its categorization, we performed the following steps:

- **Descriptive Coding.** The sources were analyzed to extract the features of MLOps Tools described that are related to the Research Questions.
- **Inter-rater Assessment.** Two researchers performed an assessment to standardize the codes extracted in a common form and ensure the completeness of the coding process, reaching consistency and rigidness between the researcher's extraction.
- **Pattern Coding.** The concept of the features extracted were grouped analyzing the relationships and creating sets of coded data.

We build our research design upon the SLR guidelines proposed in systematic literature reviews in software engineering [19]. We also used the recent grey and multi-vocal SLRs [20], [21] as a reference.

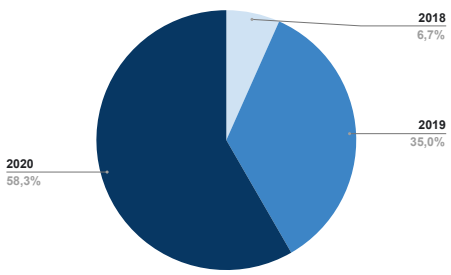

Fig. 2: Study distribution by year.

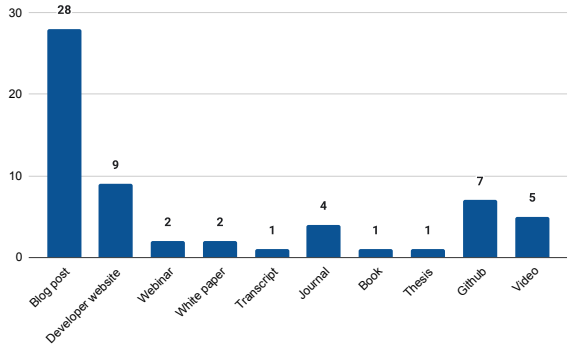

Fig. 3: Study distribution by source type.

V. Results and Discussions

Our review resulted in 60 studies. Figures 2 and 3 illustrate the distribution of the selected studies by year and by source type. The former distribution shows that contributions increase yearly and signals that MLOps has gained attention since 2018. The latter distribution shows that the selected studies are mainly from blog posts, suggesting that practitioners and researchers share their experiences, opinions, practices, and solutions concerning MLOps workflows, and tools. We report the complete list of selected studies for space limitations in our online appendix [22]. In the following, we discuss the main results and insights from our analysis.

A. RQ1 What MLOps tools can be used to build ML pipelines for Continuous Deployment?

We found 22 MLOps tools. Table I shows the complete table of MLOps tools versus the number of times mentioned by each source type. However, while exploring the online information for each tool, it became clear that not all tools are suitable for feature comparison. First, the sources do not provide the necessary information about model management in a structured way. Second, since developer websites were included in the study selection and platform/tool developers could write blog posts, the information could be biased out of self-interest. Therefore, we added two exclusion criteria to ensure the quality of the comparison:

Exclusion criteria:

5) The MLOps tool does not provide the documentation or does not list characteristics.
6) The MLOps tool is mentioned less than five times by the selected studies.

Based on the former exclusion criterion, we removed Argo (Arg) from the list of MLOps tools because we could not find any documentation. The second requirement is needed to compare only the most commonly used MLOps tools. The threshold was chosen by analyzing the number of occurrences of the MLOps tools in the selected studies. First, we excluded the three most occurring MLOps tools since they were mentioned significantly more than the others. Then, we used the average value (i.e., five) as a threshold for reference quantity. This constraint excluded eight MLOps tools with low usage: Algorithmia (Alg), Allergor (All), BentoML (Ben), Cloudera (Clo), Gradient (Gra), IBM Watson ML (IBM), Spell (Spe), and Verta (Ver). This filtering resulted in 13 MLOps tools:

Apache Airflow (Air): an open-source platform to programmatically develop, schedule, and monitor workflows as Directed Acyclic Graphs to create task instances and workflows. The pipelines are implemented as Python scripts, representing the graph structures (i.e., tasks and dependencies) as code.

AWS SageMaker (AWS): a cloud ML platform integrated into Amazon Web Services. It enables developers to create, train, and deploy ML models in the cloud through Jupyter Notebooks. It comes with built-in ML algorithms, debuggers, and utilities encapsulated in Docker containers deployed on Amazon Elastic Container Service (Amazon ECS). The training data and resulting model are stored through an Amazon Simple Storage (Amazon S3) bucket.

AzureML (Azu): a Microsoft cloud-based environment can be used to train, deploy, automate, manage, and track ML models. It enables automated model training and tuning using the Azure SDK. Azure CLI allows for developing, packaging, and validating ML models. Finally, Azure DevOps can manage deployment and automate pipelines.

DotScience (Dot): an enterprise-grade deployment and collaboration platform for MLOps. The Hub is a central workspace repository used to store the project metadata, which can be accessed through the DotScience cloud, on-premise, and in a hybrid form.

Gitlab (Git): an integrated software development platform enabling DevOps. GitLab can be installed in most GNU/Linux distributions and several cloud providers. The pipelines are configured using YAML files that define the pipeline structure, dependencies, stages, and jobs to be executed.

Google AI Platform (GAI): a platform where MLOps workflows are orchestrated as a pipeline on Google Cloud. It employs TensorFlow Extended and Kubeflow to develop pipelines built, tested, and deployed using Google Cloud Platform, which features Google Kubernetes Engine.

Jenkins (Jen): a self-contained, open-source automation server that can be used to automate tasks related to building, testing, delivering, and deploying software. It can be used as a standalone application within a built-in Java servlet container/application server as an isolated environment, installed using a Blue Ocean Docker image.

TABLE I: MLOps tools and number of times they were mentioned by source types.

Source Type		MLOps Tools																					
		TFX	*Kub*	*Air*	*MLF*	*Pol*	*Azu*	*GAI*	*Val*	*Git*	*Jen*	*Gra*	*Clo*	*AWS*	*Spe*	*Arg*	*Sel*	*Ben*	*Ver*	*IBM*	*Dot*	*Alg*	*All*
Scholar	*Journal*	0	0	0	0	0	3	3	0	0	1	0	0	3	0	0	0	0	0	1	0	0	0
	Book	1	1	1	1	0	1	1	0	1	1	0	1	1	0	0	1	0	0	0	0	0	0
	Thesis	0	1	1	1	1	1	1	0	0	1	0	0	1	0	1	0	0	1	1	0	0	0
Google	*Blog post*	8	13	5	10	2	4	4	2	4	3	1	2	6	0	2	2	0	0	0	5	1	0
	Dev. website	1	2	0	1	0	1	2	2	0	0	1	0	1	2	0	0	0	0	1	1	2	1
	Webinar	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	White paper	0	1	1	0	0	2	1	1	0	0	0	0	1	0	0	1	0	1	1	1	1	0
	Transcript	0	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	Github	3	3	1	0	1	5	2	1	0	1	0	0	2	0	1	3	2	0	0	0	0	0
	Video	2	3	1	2	0	3	0	0	2	2	0	0	1	0	1	0	0	0	0	0	0	0
	Total	**15**	**26**	**11**	**15**	**5**	**22**	**14**	**6**	**7**	**10**	**2**	**3**	**16**	**2**	**5**	**7**	**2**	**2**	**4**	**7**	**4**	**1**
	Documentation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Kubeflow (Kub): a ML toolkit to deploy ML pipelines in Kubernetes. The deployment of the workloads can be done locally, on-premises, or in cloud environments. Each step of the ML pipeline adapts the configuration choosing the platform and service required. The components are Docker images that perform a step of the pipeline.

MLflow (MLF): an open-source platform for managing the end-to-end ML lifecycle. It consists of four components: Tracking, Projects, Models, and Model Registry. The former logs parameters, code versions, metrics, and output files to compare model performance.

Polyaxon (Pol): an open-source platform to manage ML lifecycles and applications built as JSON APIs. It relies on Kubernetes to create repeatable and portable deployments with pipelines represented as DAGs. It can be executed in the cloud and on-premise, accessible via a command line interface, a dashboard, SDK, and Webhooks.

Seldon Core (Sel): an open-source platform to deploy ML models on Kubernetes. The components are wrapped up as containers. The Seldon Deployment files (i.e., in JSON or YAML) define the components, images, resources, and interactions that represent a service graph. Seldon Core leveraged Kubernetes to deploy ML models and return predictions.

TensorFlow Extended (TFX): a Google-production-scale ML platform based on TensorFlow. It provides a configuration framework and shared libraries to integrate pipeline components to define, launch, and monitor ML systems. A pipeline typically consists of eight components that handle the process from data ingestion to deployment. Each component consists of three main parts: a driver, a publisher, and an executor. The driver sends the metadata to the executor, while the publisher accepts the results of the executor and stores them in metadata.

Valohai (Val): a predictive model lifecycle management platform to build ML systems. A configuration file in YAML defines the execution steps within the project context. Each step is defined by commands executing in containers, encapsulating the main tools, libraries, and frameworks.

B. RQ2: What are the main features offered by MLOps tools?

The MLOps features we found support ML pipelines; therefore, we include all the characteristics that focus on the process, from data extraction to model monitoring. Overall, we found 32 principal features, provided in Table II, by analyzing the online documentation of the MLOps tools. Such features can be split into three categories:

- **General features** related to all phases of ML pipelines, focusing on the main characteristics of MLOps tools.
- **Data management features** related to ingesting, analyzing, transforming, storing, and monitoring the required data to ensure accessibility and reliability of ML models.
- **Model management features** related to development, delivery, evaluation, and monitoring of ML models.

1) General Features: The general features were selected based on the observed occurrence of these topics within the online documentation of the selected sources. Table III shows that MLOps tools offer similar features to deploy ML models into production. However, the approaches to implement the features differ. In essence, we could distinguish three types of MLOps tools:

- **Cloud-based ML platforms**: cloud environments comprising all steps required by ML pipelines from data storage to automated deployment (i.e., AWS Sagemaker, AzureML, Dotscience, and Google AI platform).
- **Orchestration platforms**: platforms to manage end-to-end ML pipelines relying on other resources and services for computational power (i.e., Apache Airflow, Jenkins, Kubeflow, MLflow, Polyaxon, Seldon Core, and Valohai).
- **TensorFlow Extended**: a configuration frameworks with shared libraries integrating components to define, run, and monitor ML systems.

When using CLOUD-BASED ML PLATFORMS, users must use predefined cloud services. Such platforms are easier to deploy, maintain and monitor. The solutions are more elastic and scalable to customer needs. The most common libraries are supported out-of-the-box. The main advantage of ORCHESTRATION PLATFORMS is adapting hosting services and libraries and mounting data volumes. They are highly customizable since their components are loosely coupled, self-contained, and encapsulated (e.g., using Docker). They mainly depend on cloud services for data storage and computational power, which depend on the number of training units and resource usage. TENSORFLOW EXTENDED (TFX) can be considered orchestration platforms built upon TensorFlow libraries and running on Apache Beam for data processing. However, TFX pipelines rely only on TFX components being easier to handle since all components smoothly integrate. TFX can be

TABLE II: Description of the features of MLOps tools.

Feature Category	Feature	MLOps tools
General Features	Open source	The software code is public and available for use, modification, and distribution.
	Scalability	The ability to increase workload size within existing infrastructure (hardware, software, etc.) without impacting performance.
	Elasticity	The ability to grow or shrink infrastructure (computing) resources dynamically as needed to adapt to workload changes in an autonomic manner.
	Cloud agnostic/native	The performance is consistent regardless of what platform it's deployed on. In this research this constraint is met if it could be deployed on: Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform (GCP).
	Extensibility	Easily define your own operators, executors and extend the library so that it fits the level of abstraction that suits your environment.
	Metadata management/collection	Metadata management is used to collect data during the complete ML pipeline.
	Isolation/loosely coupling	Components can be developed and deployed independently and depend on each other to the least extent practicable.
	CI/CD	The platform supports Continuous Integration (CI) and Continuous Delivery (CD) for the complete ML pipeline.
	UI	User Interface or Dashboard.
	CLI	Command Line Interface.
	API gateway	Instead of calling services directly, clients can call the API gateway,which forwards the call to the appropriate services on the back end and services as entry point for clients.
	DAGs	Directed Acyclic Graphs are used to describe the workflow or can be encapsulated within the platform.
Data Management Features	Data streaming (real-time)	Continuous flow of data generated by various data sources is supported and can be processed, stored, analyzed, and acted upon directly.
	Data storage	A build-in database to store raw data, projects and metadata.
	Data analysis	A pipeline component generates features statistics over both training and serving data, which can be used by other pipeline components.
	Data transformation	A pipeline component identifies anomalies in training and serving data and prepares the data for ML tasks. The output of this step are the data splits.
	Data monitoring	The data is monitored to maintain the quality and inspect general metrics.
	Metadata management/collection	Metadata management is used to collect data and can be used to determine which data is used to train a model.
	API endpoint	The output of the data management can be called by using a API gateway, which forwards the data, metadata or data schema.
	Automation	The data management process can be automatically executed in production based on a schedule or in response to a trigger.
Model Management Features	Library agnostic/native	All (top) ML frameworks and libraries are supported.
	Model tracking	Intermediate ML model performance can be tracked and logged to maintain reproducibility and gain insight.
	Model registry	A centralized repository used to standardize the definition, storage,and access of features for training and serving which is accessible via an API.
	Hyper parameter tuning	An optimization engine is encapsulated for hyper-parameter tuning to efficient train the ML models.
	A/B testing	A/B testing can be used to track differences between two predictive model versions or models can run in parallel on different end-points.
	Anomaly detection	Outliers are automatically identified to uncover irregular patterns of the ML model.
	Drift detection	Significant changes in data distributions and the prediction performance are automatically detected to prevent staleness and accuracy decrease.
	Threshold alert	It is possible to set up alerts when prediction distribution varies significantly from expected values.
	Performance monitoring	The model predictive performance is monitored to potentially invoke a new iteration in the ML process.
	Metadata management/collection	Metadata management is used to record ML model, the performance and runtime parameters.
	API endpoint	The output of the model management can be called by using a API gateway, which forwards the model predictions.
	Automation	The model management process can be automatically executed in production based on a schedule or in response to a trigger.

integrated into other environments and orchestration platforms (e.g., Kubeflow and Apache Airflow).

2) Data Management Features: Table IV lists the features related to data management features for each MLOps tool. All cloud-agnostic MLOps tools support data ingestion of binary data sources hosted in the cloud. Acquiring new data without hosting could services is not supported. Besides common cloud-based platforms, MLOps tools supporting data storage (e.g., Dotscience, Polyaxon, and Valohai) also allow for building data stores that provide direct access to raw data. This service integrates file tracking systems to monitor data updates constantly. Data is not parsed after every execution but specific artifacts that record and retrieve metadata and data schemes associated with the dataset(s) are created during the first run of the pipeline. Additional statistics about the dataset are extracted to validate incoming data, transform data, and perform feature engineering operations. The result is a saved model including the final train (and test) data, either temporally stored or used in the cloud storage. Cloud providers can also handle the data management process (e.g., AWS

TABLE III: General features of MLOps tools.

Feature	MLOps tools												
	Air	AWS	Azu	Dot	Git	GAI	Jen	Kub	MLF	Pol	Sel	TFX	Val
Open Source	x	-	-	-	-	-	x	x	x	x	x	x	-
Scalability	x	x	x	x	x	x	x	x	x	x	x	x	-
Elasticity	-	x	x	-	x	x	x	x	x	x	-	-	x
Cloud Agnostic	x	-	-	x	x	-	x	x	x	x	x	x	x
Extensibility	x	x	x	x	-	x	-	x	x	x	x	-	x
Metadata Collection	x	x	x	x	x	x	x	x	x	x	x	x	x
Isolation	x	x	x	-	-	x	-	x	x	x	x	x	-
CI/CD	-	-	x	x	x	x	x	-	x	-	x	-	-
UI	x	x	x	x	x	x	x	x	x	x	x	-	x
CLI	x	x	x	x	x	x	x	x	x	x	x	x	x
API Gateway	x	x	x	x	x	x	-	x	x	x	x	x	x
DAGs	x	-	-	-	x	x	x	x	x	x	x	x	-

TABLE IV: Data management features of MLOps tools.

Feature	MLOps tools												
	Air	AWS	Azu	Dot	Git	GAI	Jen	Kub	MLF	Pol	Sel	TFX	Val
Data Streaming	x	x	x	-	x	x	-	x	x	x	x	-	x
Data Storage	-	x	x	x	-	x	-	-	-	x	-	-	x
Data Analysis	x	x	x	x	x	x	x	x	x	x	x	x	x
Data Transformation	x	x	x	x	x	x	x	x	x	x	x	x	x
Data Monitoring	-	x	x	x	x	x	x	x	-	x	-	x	x
Metadata Collection	-	-	x	x	x	x	x	x	x	x	x	x	-
API endpoint	-	x	x	-	x	x	x	-	-	x	-	x	x
Automation	x	x	x	x	x	x	x	x	x	x	x	x	x

Glue). Extract, transform and load (ETL) services register the data sources, extract data from the sources, store the associated metadata, transform (e.g., cleaning) them to match target schemas, and load them into databases. Such a process is automated and requires low effort to build, maintain, and run components. However, these services can cause problems when on-demand/real-time transformations are needed. Furthermore, when the process is handled by external services not included in the MLOps pipeline, the training-serving skew could be hard to manage. Nevertheless, in all cases explained above, the data management process is similar using the different MLOps tools.

TABLE V: Model management features of MLOps tools.

Feature	MLOps tools												
	Air	AWS	Azu	Dot	Git	GAI	Jen	Kub	MLF	Pol	Sel	TFX	Val
Library Agnostic	x	x	x	x	-	x	-	x	x	x	x	-	x
Model Tracking	x	x	x	x	x	x	-	x	x	x	x	x	x
Model Registry	x	x	x	x	x	x	x	x	x	x	x	x	x
Hyperparameter Tuning	x	x	x	x	-	x	-	x	x	x	-	x	x
A/B Testing	-	x	x	x	-	-	-	x	x	-	x	x	-
Anomaly Detection	x	x	x	x	x	-	-	-	-	-	x	-	-
Drift Detection	-	x	x	x	-	x	-	-	-	-	x	x	-
Threshold Alert	x	x	x	x	x	-	-	-	-	-	-	-	-
Performance Monitoring	x	x	x	x	x	x	-	x	x	x	x	x	x
Metadata Collection	-	x	x	x	x	x	x	x	x	x	x	x	x
API endpoint	x	x	x	x	x	x	-	x	x	x	x	x	x
Automation	x	x	x	x	x	x	x	x	-	x	x	x	x

3) Model Management Features: Table V lists the features for model management of each MLOps tool. After consuming training data, evaluation data, and necessary metadata, ML models can be deployed for the first time. Such models are trained and validated until the requested performance is reached to consider the model publishable. Please consider that this goal could be reached even without MLOps. MLOps foresees the continuously delivering of prediction services based on newly trained models; therefore, model management needs to be automated using model validation steps, pipeline triggers, and metadata management. All considered MLOps tools support automated executions based on triggers (e.g., Github push) or scheduled time windows with new models tracked and saved in model registries. Therefore, we conclude that all tools allow for a high level of automation and continuous delivery of new models, which eventually can be queried with API prediction services. A/B testing is a powerful method to identify the best models before deployment by allowing users to compare their predictions. Performance must be monitored to obtain reliable continuous delivery. The executions can be recorded, analyzed, and monitored through metadata management. If this process automatically triggers new executions, the model management process can be fully automated, significantly reducing human effort. Therefore, MLOps tools should provide optimization engines that automatically search for the best hyper-parameter settings, detect drifts and anomalies and eventually alert the user if necessary. This high maturity level of an ML process can be reached only by three MLOps tools: AWS Sagemaker, Azure ML, and Dotscience. In the long term, some manual steps are still needed to verify the model compatibility with the target infrastructure and resolve issues. Developing a fully automated model management process is challenging, and it is recommended to gradually implement each step before aiming at full automation. The selected MLOps tools allow testing and deploying new models in a (semi-)automated fashion while governing the single execution steps.

VI. Threats to Validity

Our study is composed of two main parts: the multivocal literature review, and the feature analysis. This section describes the main limitations and threats to validity of both of them.

a) Multivocal Literature Review: The main threats affecting the validity of our MLR are related to the way we selected and extracted the data used for our analyses. The most crucial challenge for any literature review is identifying an effective and complete set of search terms. To mitigate this threat, we defined a detailed search strategy. Once we extracted the first set of keywords, we identified synonyms or alternative spellings, and then we verified the presence of such terms in the relevant papers. Moreover, the second author of this paper double-checked the steps leading to selecting the relevant papers. Finally, we applied exclusion and inclusion criteria to extract our interest materials. The research resulted in 60 studies, including also blog posts, allowing us to study both practitioners and researchers perspective. Although our search query focused on the concept of MLOps, this definition is recent; therefore, it could be partially used in grey literature, thus possibly implying that some sources of literature were not considered during the work due to related terms. Furthermore, the research methodology doesn't include the snowballing inside the process. This missing step could also help retrieving more sources of literature in order to go deeper in the review.

b) Feature Analysis: Other threats are related to the meta-analysis performed when comparing the different MLOps tools extracted. Feature analysis and comparison are

based solely on the documentation of the MLOps tools. We distinguished between data management and model management. However, these concepts are strongly related and intertwined for model lifecycle support. In particular, cloud-orchestration platforms cover these processes together. Furthermore, to fully capture and analyze the features of MLOps tools, solely documentation is not sufficient, and they should be evaluated in production.

VII. Conclusion and Future Work

In this work, we performed a MLR to list the most common MLOps tools and the features they implement. As a result, we extracted 13 MLOps tools, and we discovered that their implementation and integration with other services are different despite such features being similar. For example, some tools, e.g., Apache Airflow, define executions as experiments or runners, while others use Directed Acyclic Graphs to represent the tasks of the pipelines. However, all tools allow a high level of integration and configuration and can be used to build end-to-end ML pipelines and deliver ML models. In addition, the tools leverage cloud services to process data, containers to package applications, orchestration platforms to define pipelines, and dashboards to track and monitor models.

Orchestration platforms allow (and recommend) the user to combine different services. On the one hand, they allow for getting the best from each tool and service, while, on the other hand, finding the best combination could be tricky. Devising methods to foster the integration of MLOps tools is part of our future agenda. Finally, we found that companies and platforms often define MLOps differently to raise their number of customers. Therefore, we plan to provide a standard definition of MLOps as part of our future agenda.

Acknowledgments

The authors would like to thank Rutger Drubbel for initiating data collection and analysis during his Master's thesis at JADS. Fabio is partially supported by the Swiss National Science Foundation - SNF Project No. PZ00P2_186090. This work has been also partially supported by the EMELIOT national research project, which has been funded by the MUR under the PRIN 2020 program (Contract 2020W3A5FY).

References

[1] J. Humble and D. Farley, *Continuous delivery: reliable software releases through build, test, and deployment automation*. Pearson Education, 2010.

[2] M. Guerriero, M. Garriga, D. A. Tamburri, and F. Palomba, "Adoption, support, and challenges of infrastructure-as-code: Insights from industry," in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2019, pp. 580–589.

[3] F. Kumeno, "Sofware engneering challenges for machine learning applications: A literature review," *Intelligent Decision Technologies*, vol. 13, no. 4, pp. 463–476, 2019.

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

[5] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich, "Data management challenges in production machine learning," in *Proceedings of the 2017 ACM International Conference on Management of Data*, ser. SIGMOD '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1723–1726.

[6] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley, "Google vizier: A service for black-box optimization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2017.

[7] B. Benni, M. Blay-Fornarino, S. Mosser, F. Précisio, and G. Jungbluth, "When devops meets meta-learning: A portfolio to rule them all," in *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 2019, pp. 605–612.

[8] H. Atwal, *Practical DataOps: Delivering Agile Data Science at Scale*. Springer, 2019.

[9] M. Boehm, A. Kumar, and J. Yang, "Data management in machine learning systems," *Synthesis Lectures on Data Management*, vol. 14, pp. 1–173, 02 2019.

[10] K. O'Leary and M. Uchida, "Common problems with creating machine learning pipelines from existing code," 2020.

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

[12] D. Baylor, K. Haas, K. Katsiapis, S. Leong, R. Liu, C. Mewald, H. Miao, N. Polyzotis, M. Trott, and M. A. Zinkevich, "Continuous training for production ml in the tensorflow extended (tfx) platform," in *OpML*, 2019.

[13] T. Kakantousis, A. Kouzoupis, F. Buso, G. Berthou, J. Dowling, and S. Haridi, "Horizontally scalable ml pipelines with a feature store," in *Proc. 2nd SysML Conf., Palo Alto, USA*, 2019.

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

[15] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, no. 2004, pp. 1–26, 2004.

[16] V. R. B. G. Caldiera and H. D. Rombach, "The goal question metric approach," *Encyclopedia of software engineering*, pp. 528–532, 1994.

[17] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[18] J. Saldaña, *The coding manual for qualitative researchers*. sage, 2021.

[19] V. Garousi, M. Felderer, and M. V. Mäntylä, "Guidelines for including grey literature and conducting multivocal literature reviews in software engineering," *Information and Software Technology*, vol. 106, pp. 101–121, 2019.

[20] I. Kumara, M. Garriga, A. U. Romeu, D. Di Nucci, F. Palomba, D. A. Tamburri, and W.-J. van den Heuvel, "The do's and don'ts of infrastructure code: A systematic gray literature review," *Information and Software Technology*, vol. 137, p. 106593, 2021.

[21] S. Peltonen, L. Mezzalira, and D. Taibi, "Motivations, benefits, and issues for adopting micro-frontends: a multivocal literature review," *Information and Software Technology*, vol. 136, p. 106571, 2021.

[22] G. Recupito, F. Pecorelli, G. Catolino, S. Moreschini, D. Di Nucci, F. Palomba, and D. A. Tamburri, "Online appendix," 2022. [Online]. Available: https://github.com/gilbertrec/MLR-MLOps-Tools-Features-Appendix